Closed-Form Change Detection from Moving Light Field Cameras

Donald G. Dansereau[1] , Stefan B. Williams[2] , and **Peter I. Corke**[1]

[1]Australian Centre for Robotic Vision, Queensland University of Technology
[2]Australian Centre for Field Robotics, University of Sydney

IROS 2015 Workshop
Alternative Sensing for Robot Perception

Stationary Cameras Simplify Things

Denoising [Bennett2005]

Change Detection [Chien2002]

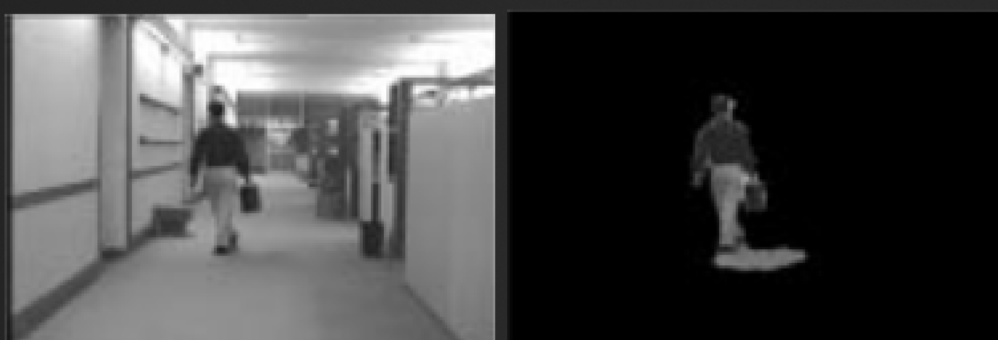
Tracking [Zhao2002]

Classic, simple video processing

- Denoising
- Change detection
- Tracking
- Segmentation
- Temporal filtering

Stationary Cameras Simplify Things

Denoising [Bennett2005]



Change Detection [Chien2002]

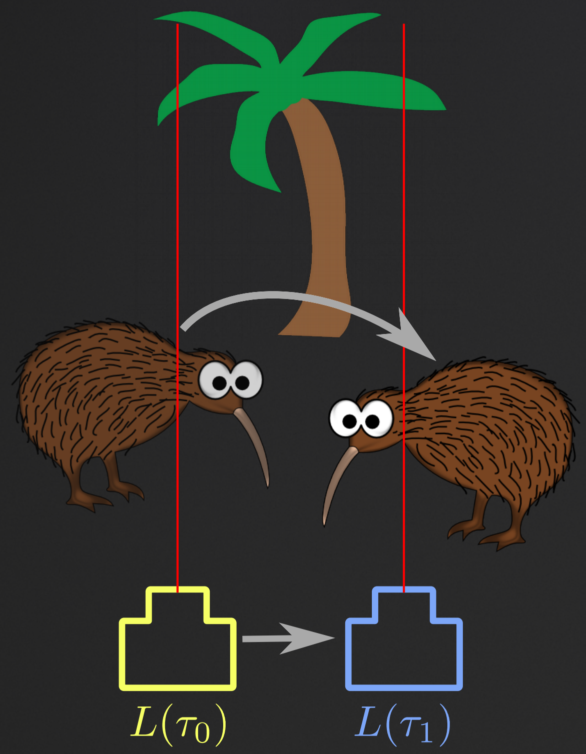
Tracking [Zhao2002]

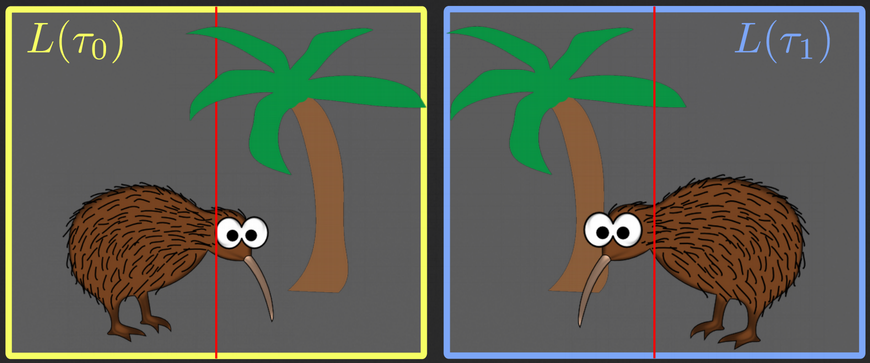
Classic, simple video processing

- Denoising
- Change detection
- Tracking
- Segmentation
- Temporal filtering

But these break
when the camera moves

The Problem



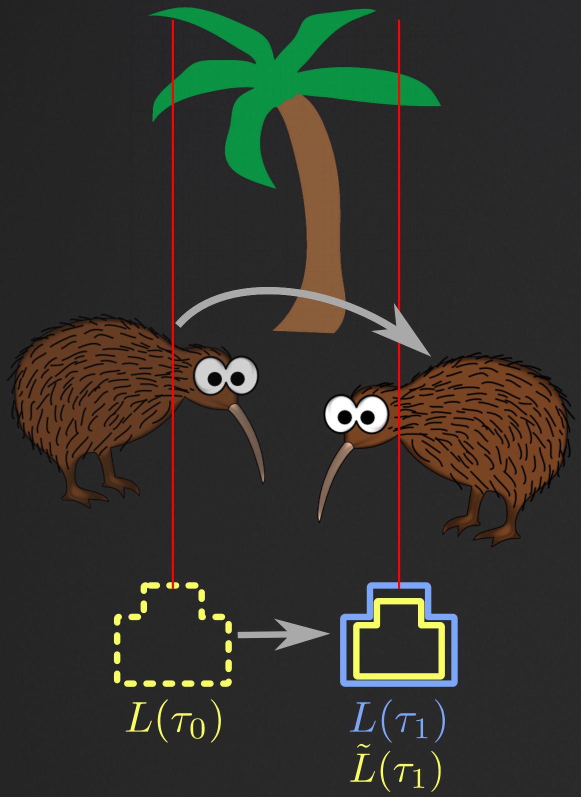


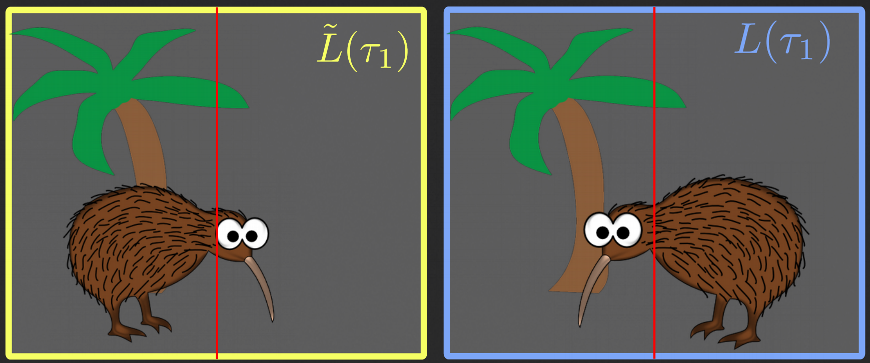
Moving camera, 3D scene

→ Nonuniform apparent motion

→ Breaks static-camera methods

A Simple Solution?



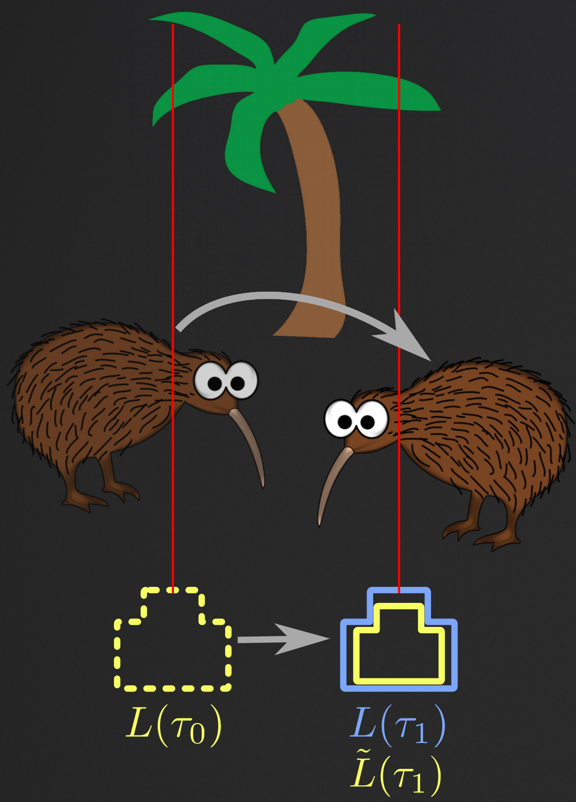


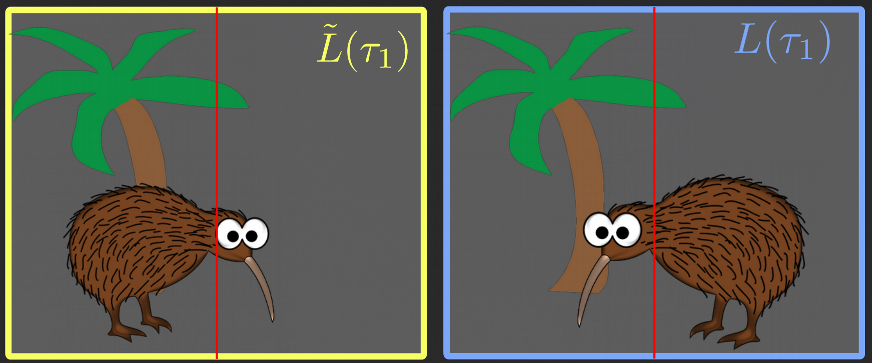
Fake a stationary camera

→ No apparent motion

→ Static-camera methods work

A Simple Solution?



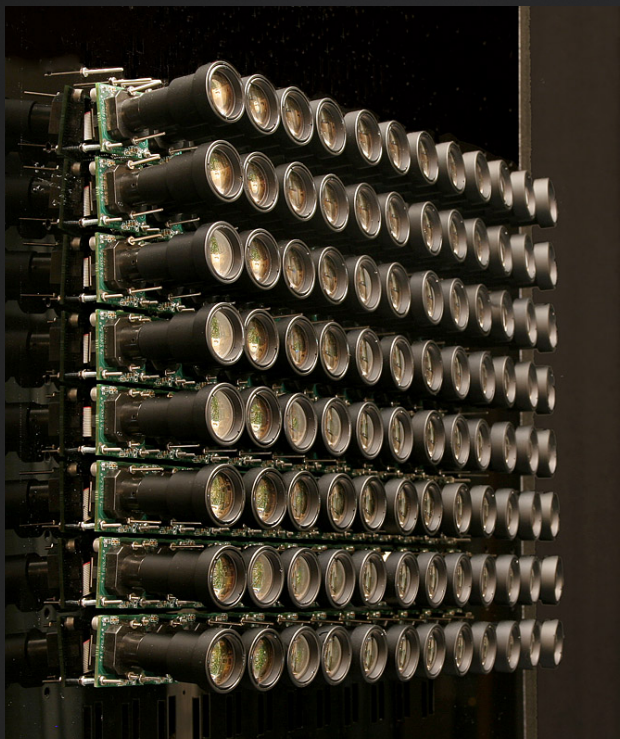


Fake a stationary camera
→ No apparent motion
→ Static-camera methods work

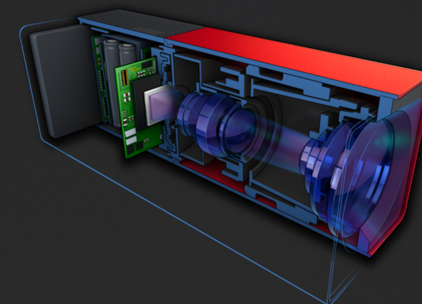
Not so simple!

- Dense structure from motion
- Iterative optimization, outlier rejection
- Complex behaviours, failure modes
- Complex to implement well

Light Field Imaging: A Quick Tour



Stanford camera array



Lytro lenslet-based cameras

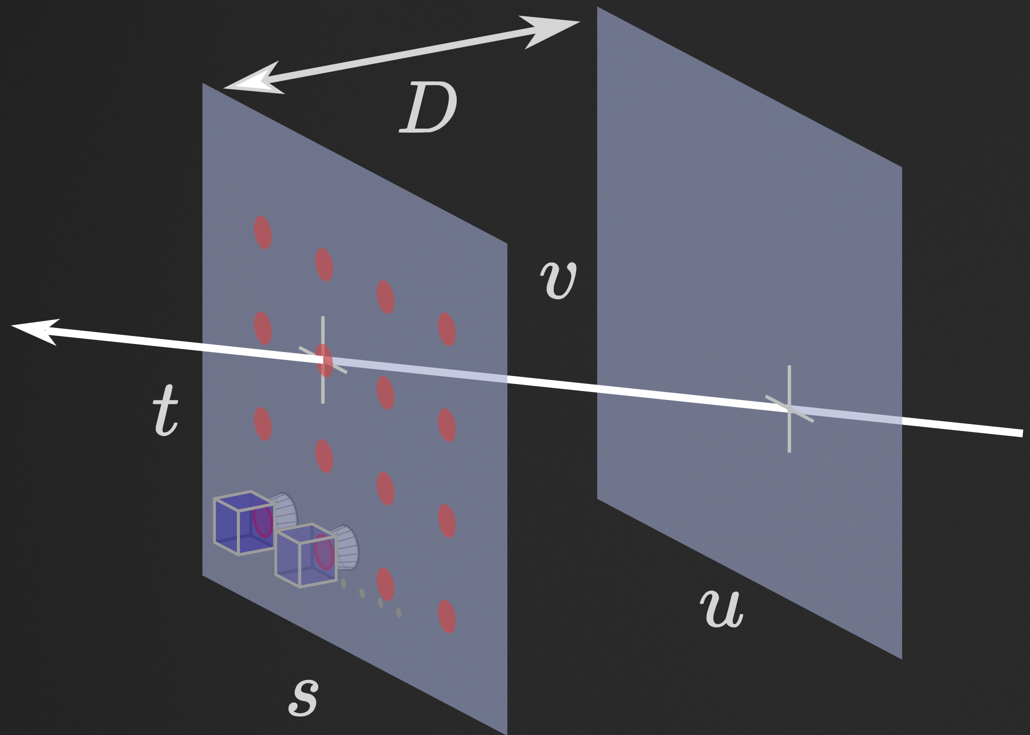
Pelican Imaging

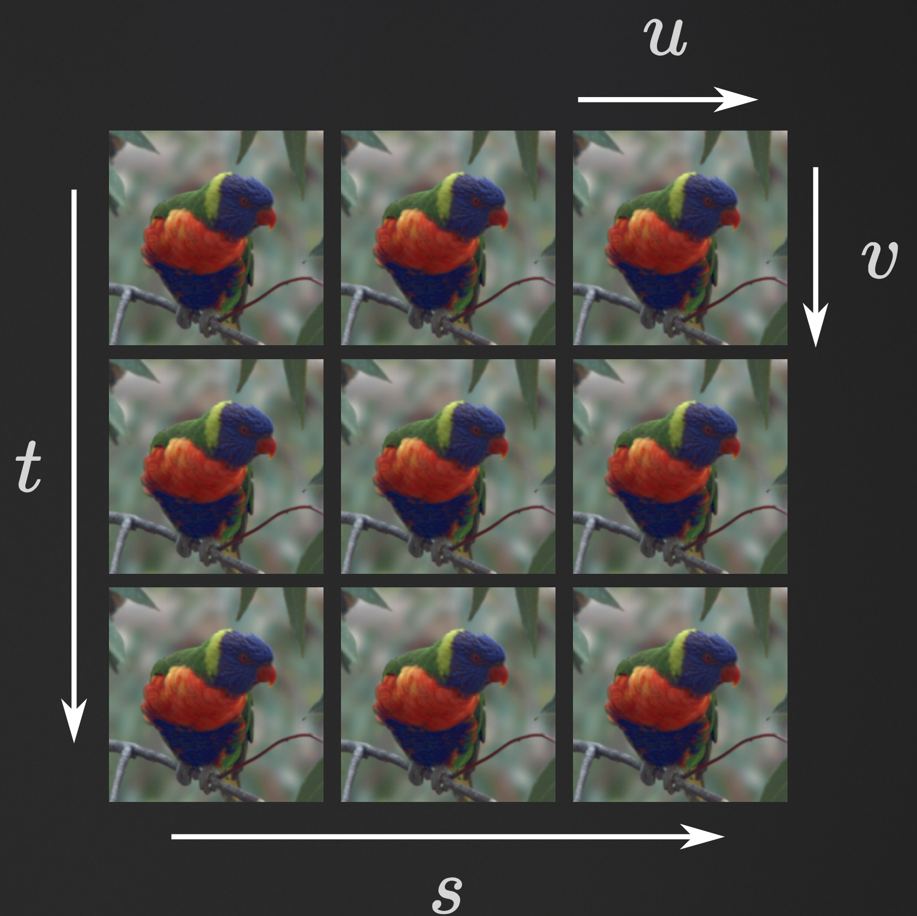
Raytrix

Linx Imaging (Apple)

- New tradeoffs → More light, more depth of field
- New image geometry → new capabilities, simplifications, robustness

The Light Field

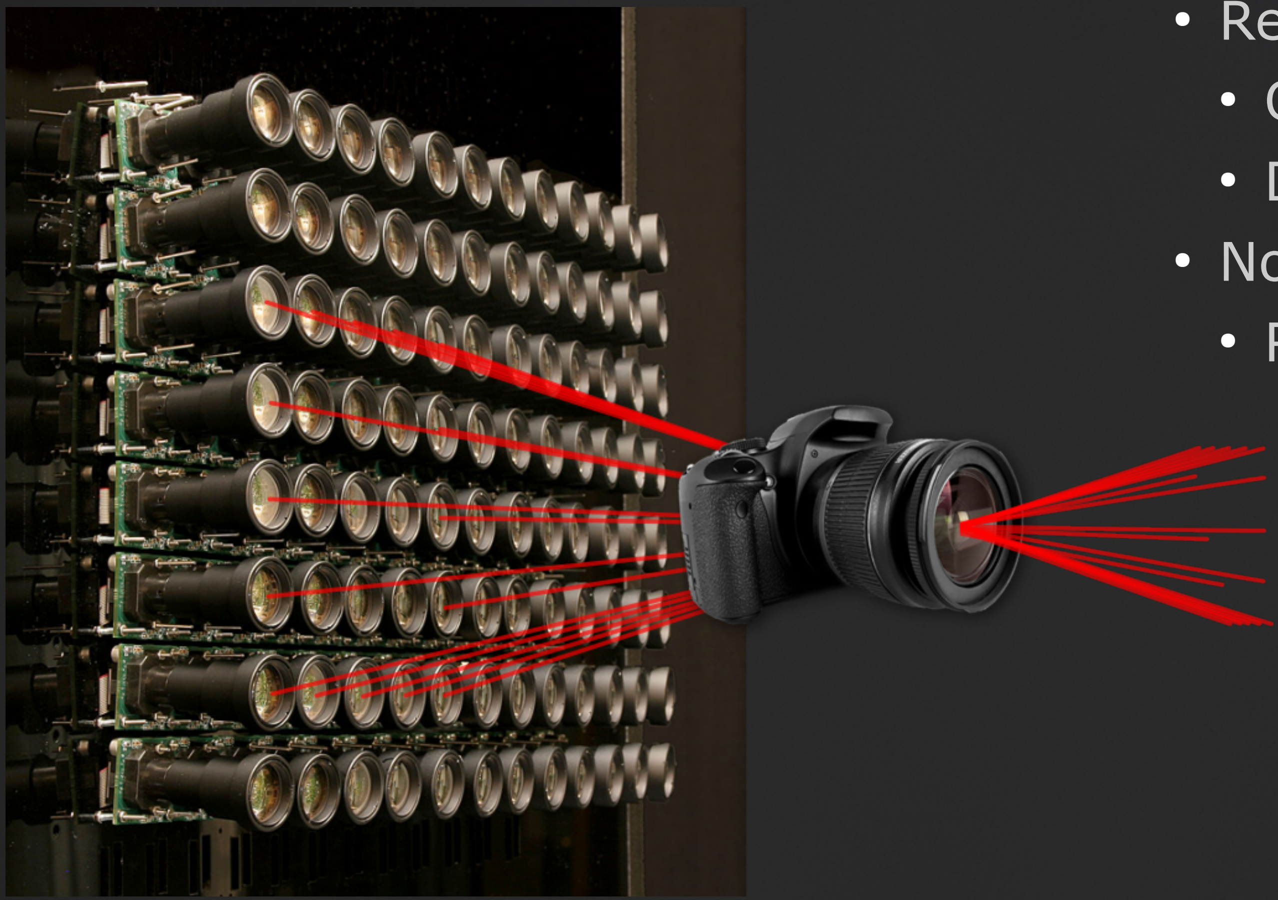




- A 2D array of 2D images
- A 4D array of pixels
- Pixels map to rays

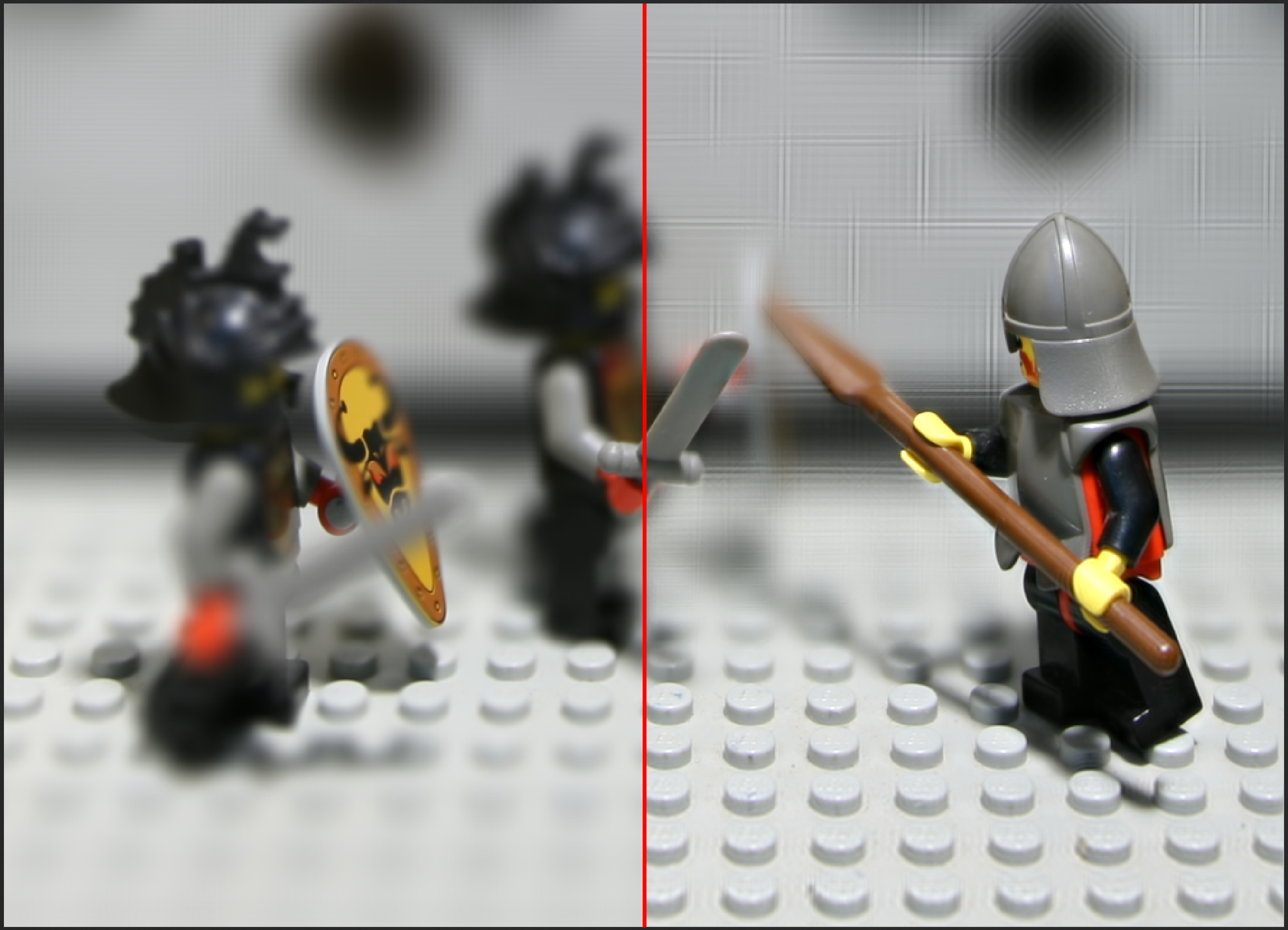
4D Image $\mathcal{L}(s, t, u, v)$

Light Field Rendering



- Render novel views
  - Off-plane
  - Different lenses
- No 3D model
  - Ray interpolation

Light Field Rendering

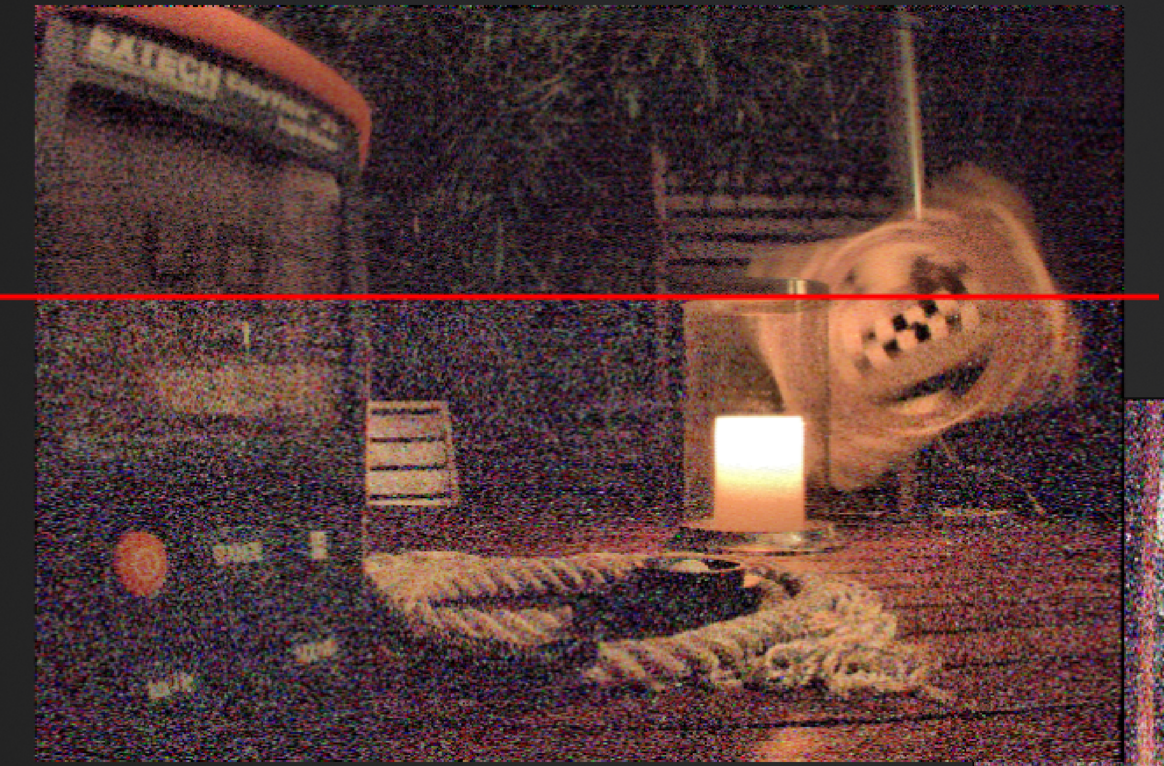


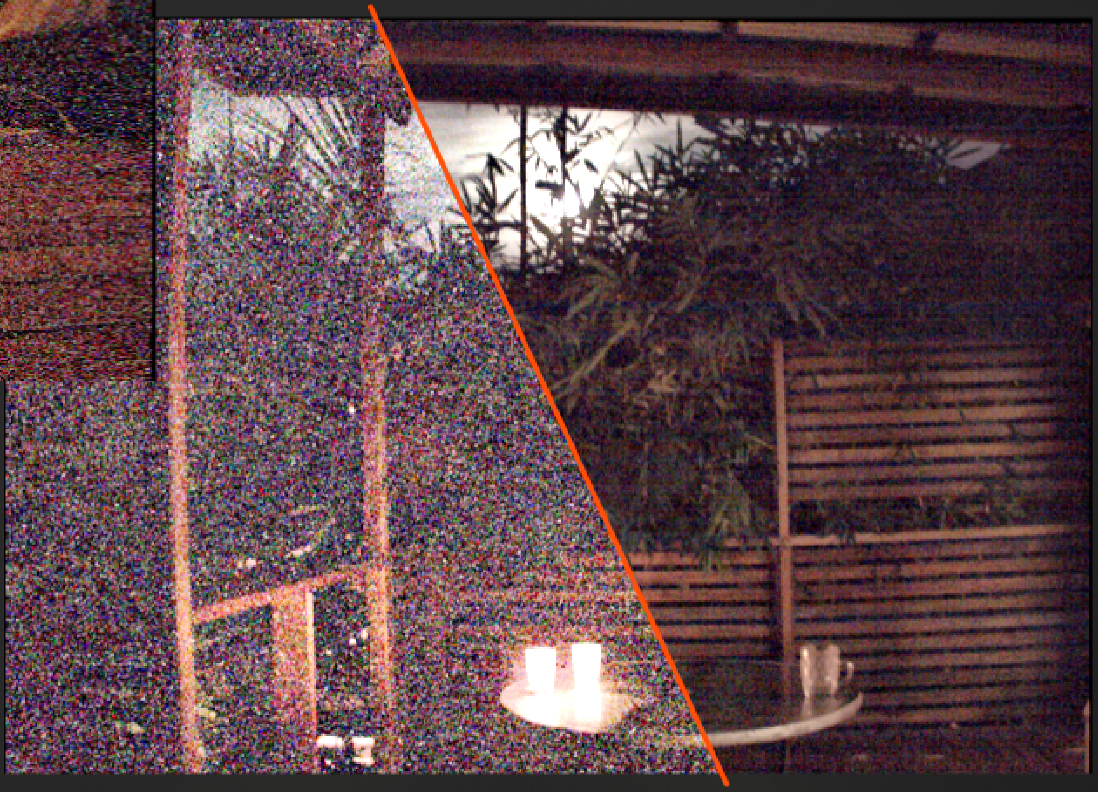
Planar refocus ... volumetric refocus

[Dansereau2015]

[LF c/o Stanford Computer Graphics Laboratory]

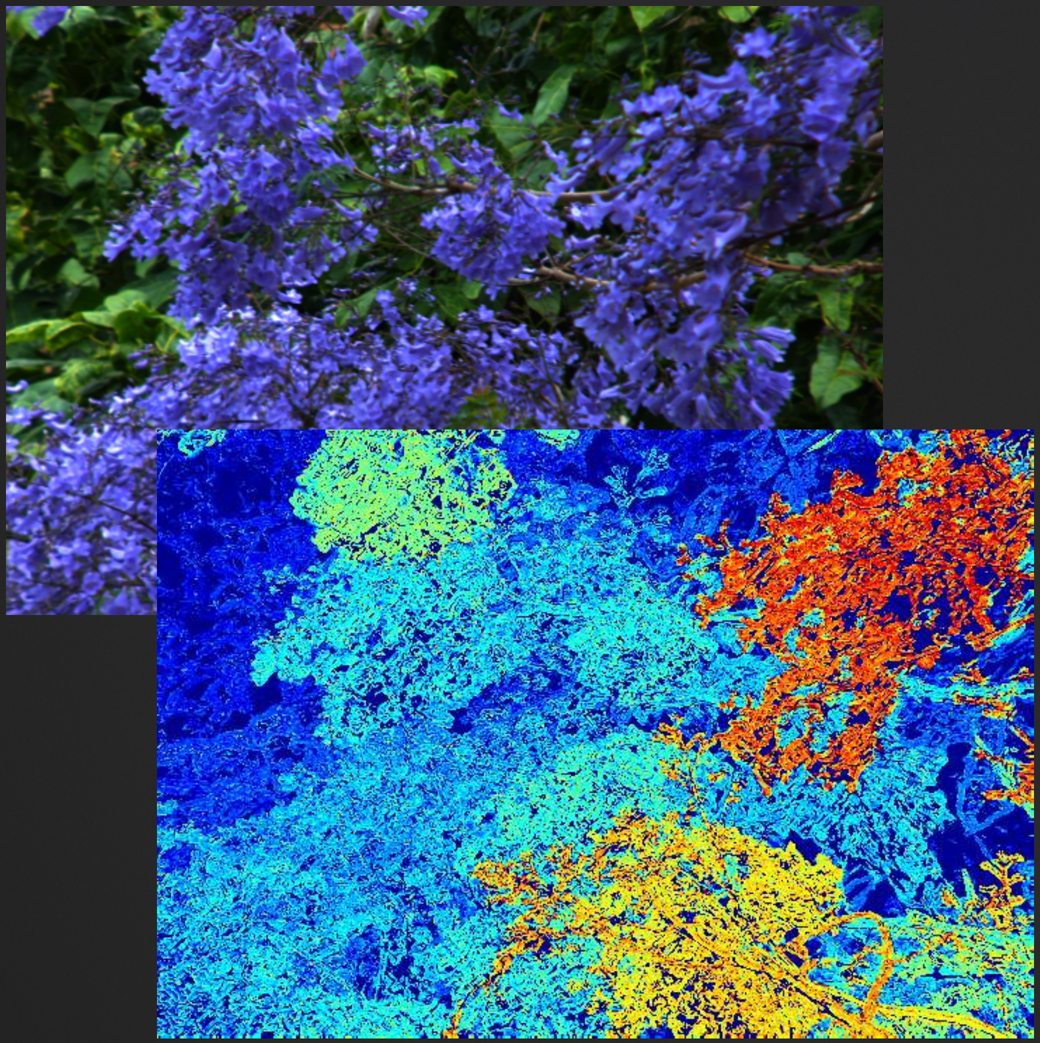
Low-Light Imaging / Denoising



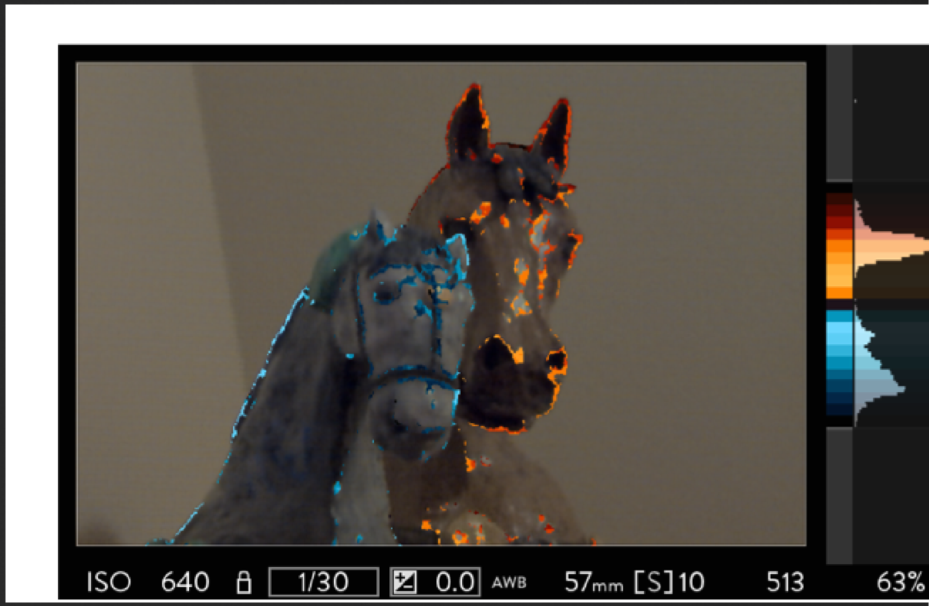


[Dansereau2015]

Depth Estimation



[Dansereau2004]

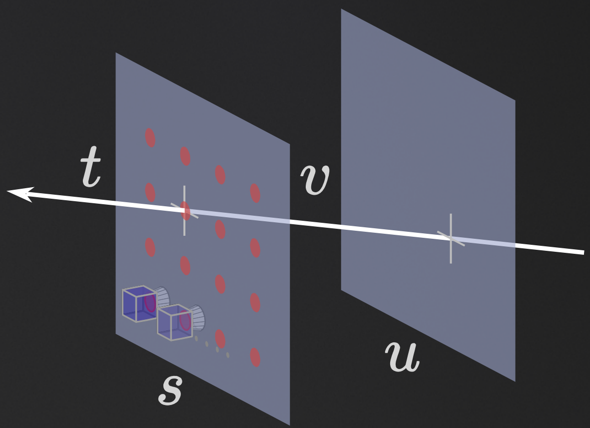


Lytro, 2014

- 4D gradients map to depths
- Real-time single-camera depth
- RGBD that works great outside

Plenoptic Flow: Velocity Estimation

- Generalized optical flow
- Built on 1st differences e.g. $L_s = L(s+1) - L(s)$
- Decomposes change into 6 components
- Closed-form least squares solution



$$\begin{bmatrix} L_s \\ L_t \\ L_z \\ (t + vL_u/L_s)L_z - DL_v \\ -(s + uL_v/L_t)L_z + DL_u \\ sL_t - tL_s + uL_v - vL_u \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} q_x \\ q_y \\ q_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = -L_\tau$$

Change over time

6 motion components

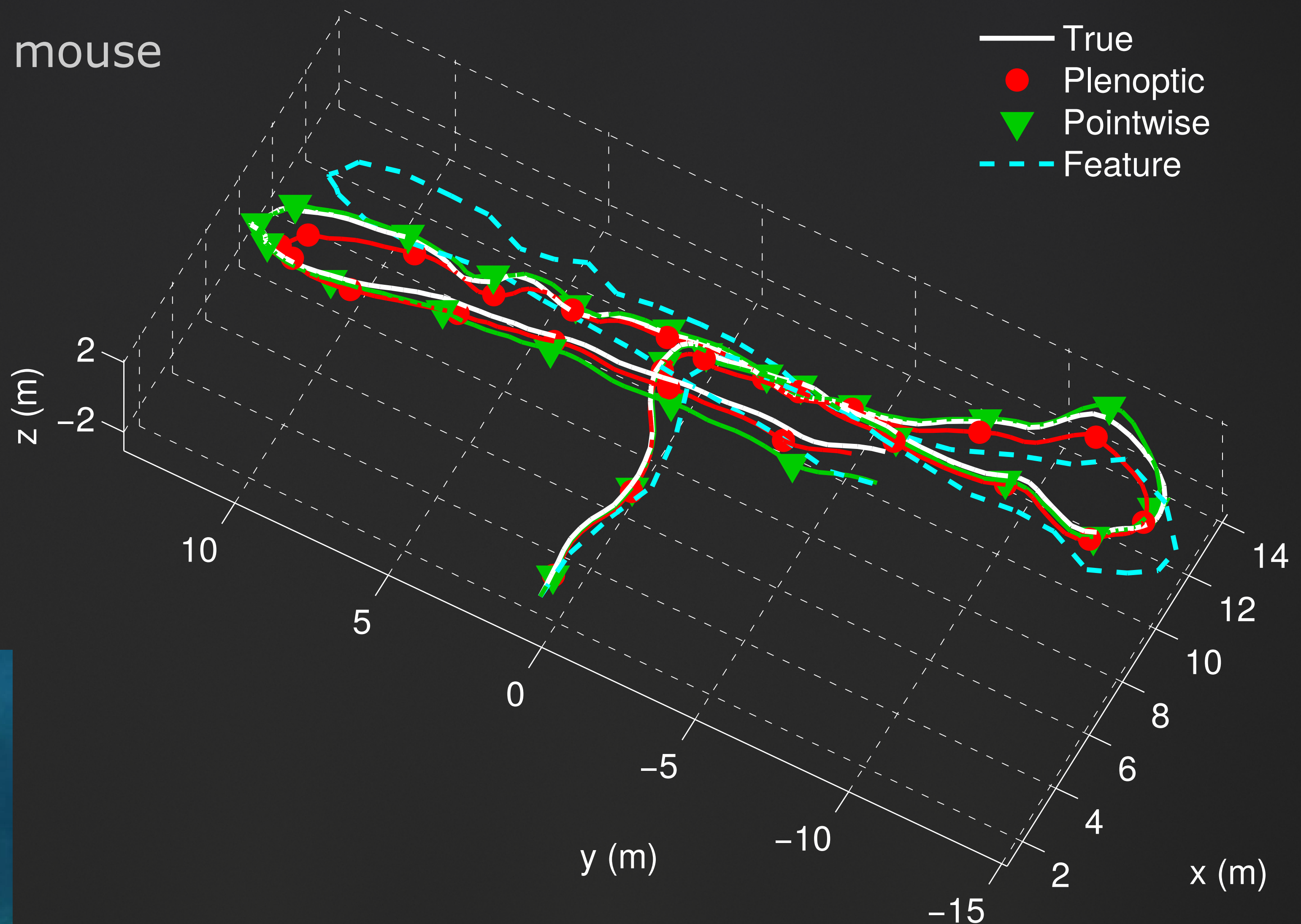
Camera rotation, translation

$$\boldsymbol{Av} = L_\tau$$

[Dansereau2011]

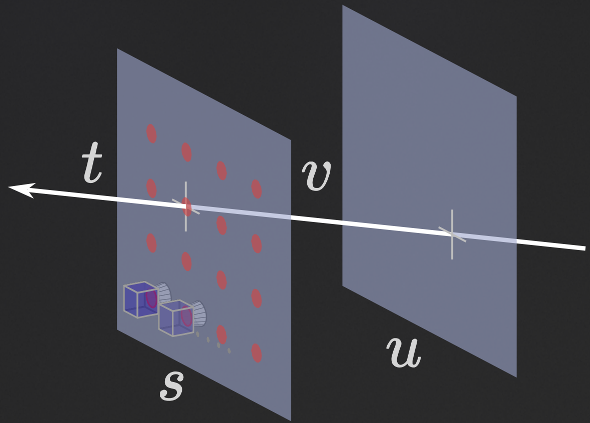
Plenoptic Flow: Velocity Estimation

- Like a 3D optical mouse

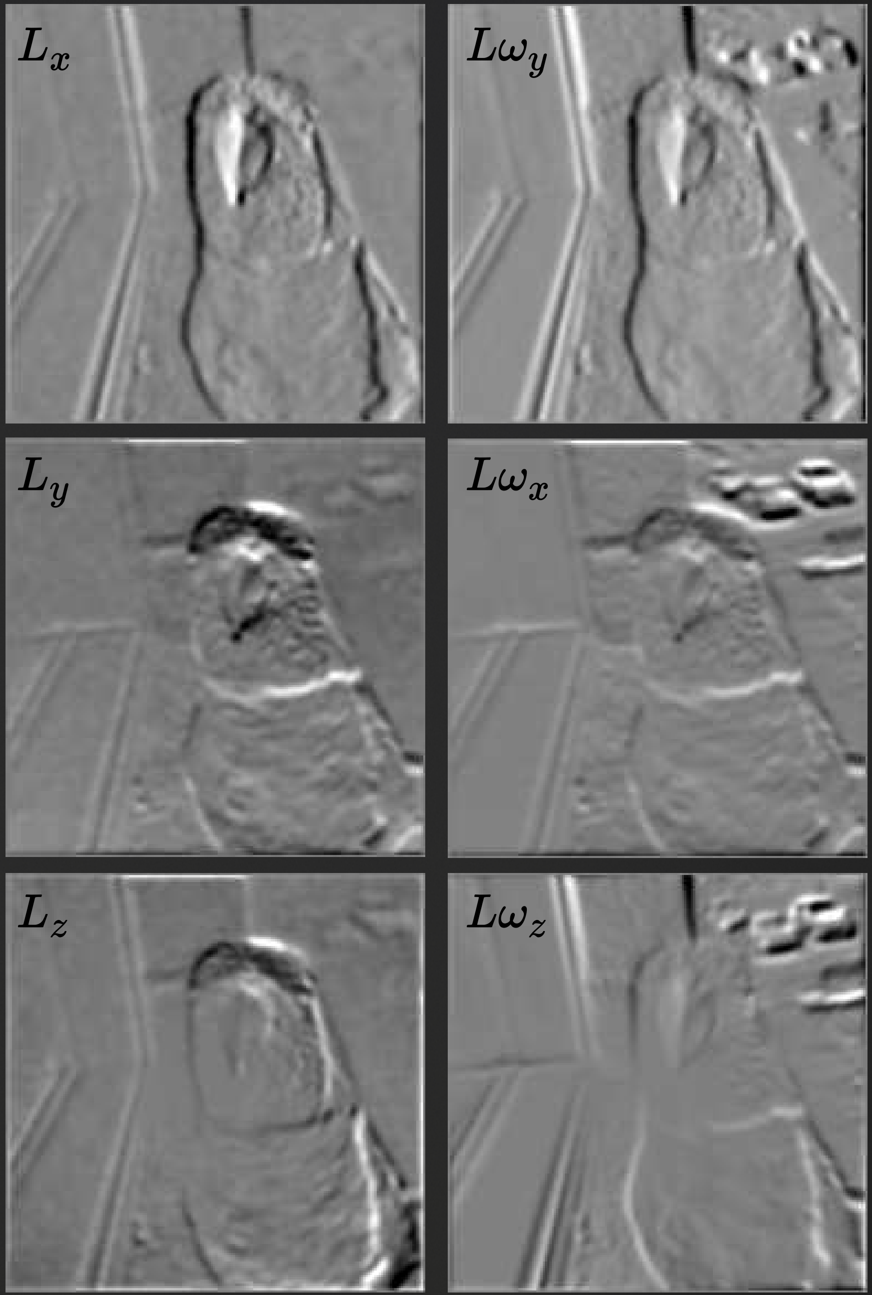


[Dansereau2011]

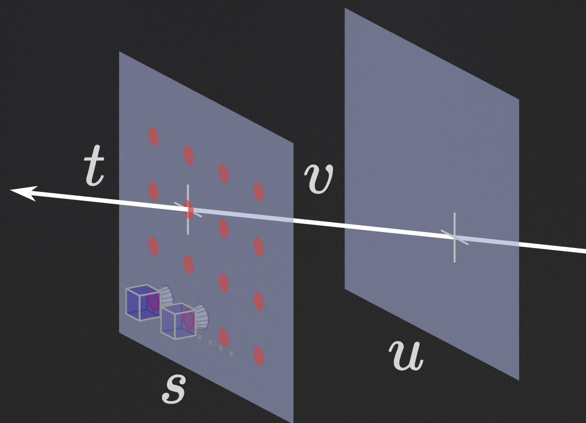
Plenoptic Flow: Additive Rendering



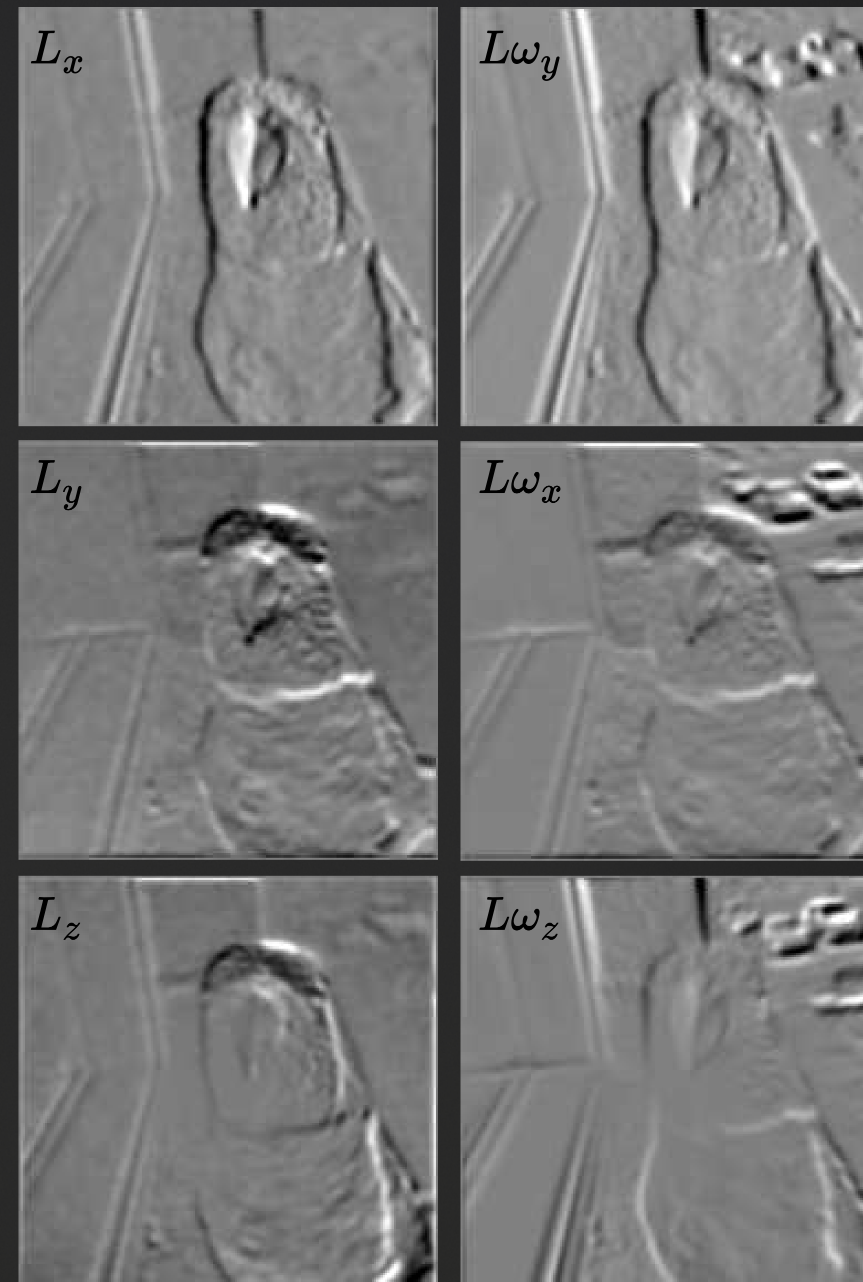
L



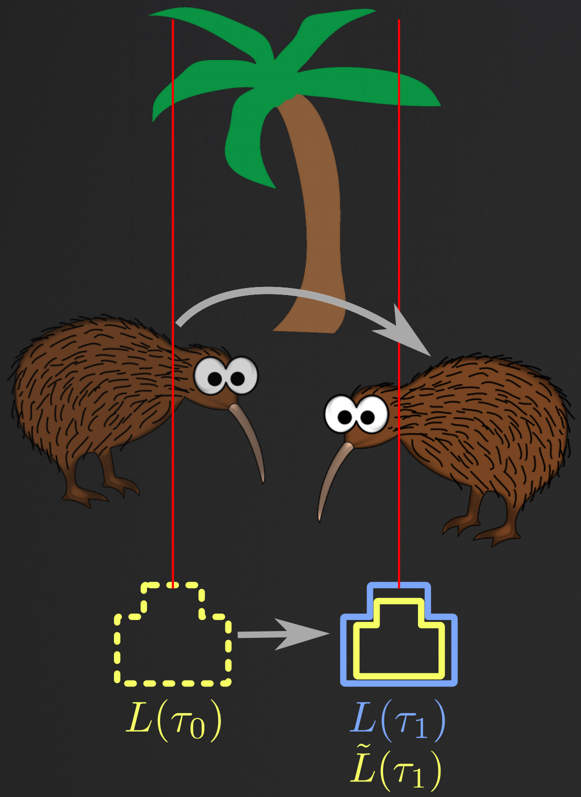
Plenoptic Flow: Additive Rendering

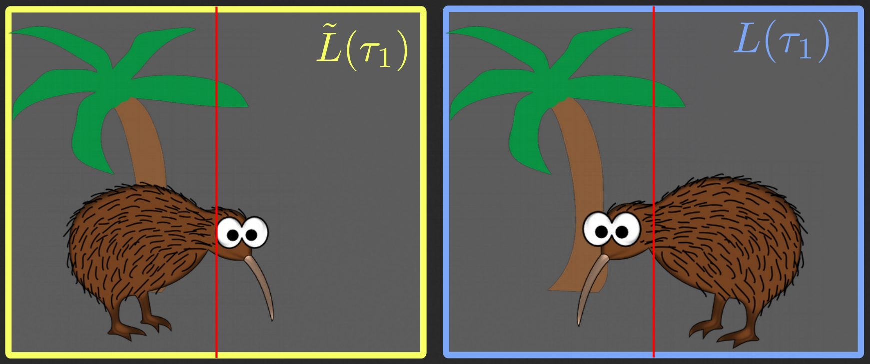


$L + kL_z$



A Simple Solution After All?



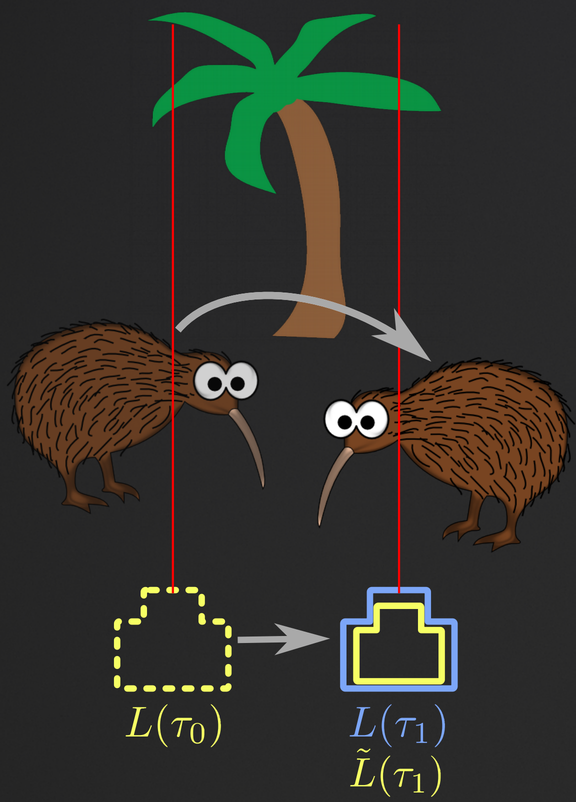


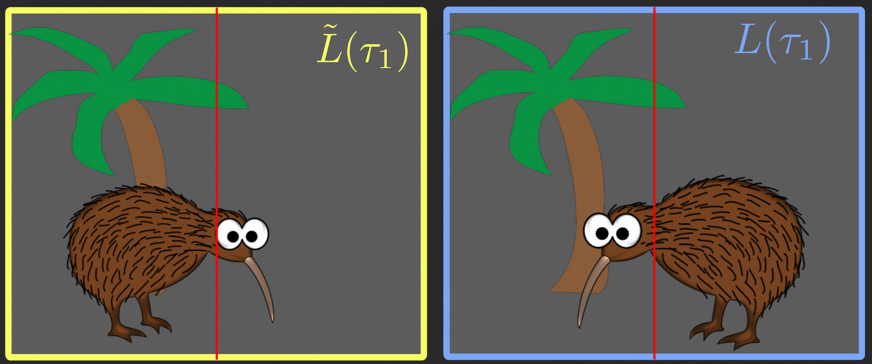
Estimate camera motion : Closed-form

Render new view : Closed-form

Simple conversion to static camera

A Simple Solution After All?





Estimate camera motion : Closed-form

Render new view : Closed-form

Simple conversion to static camera

- No 3D model
- Closed-form, constant runtime
- Simple behaviours, failure modes
- Easy to implement in parallel HW
  - FPGA, GPU, etc.

Demo: Change Detection

Fixed Camera

- Simple, robust
- Parallel

[Chien2002]

Moving Camera

- Computationally, behaviourally complex
- Iterative, nonlinear
- Sparse or constrained

[Sheikh2009]

Demo: Change Detection

Estimate camera motion
(closed-form least-squares)

$$\boldsymbol{A}\boldsymbol{v} = L_\tau \;\rightarrow\; \tilde{\boldsymbol{v}}$$

Demo: Change Detection

Estimate camera motion
(closed-form least-squares)

$$\boldsymbol{A}\boldsymbol{v} = L_\tau \rightarrow \tilde{\boldsymbol{v}}$$

Change due to camera motion

$$\tilde{L}_\tau = \boldsymbol{A}\tilde{\boldsymbol{v}}$$

Demo: Change Detection

Estimate camera motion
(closed-form least-squares)

$$\boldsymbol{A}\boldsymbol{v} = L_\tau \rightarrow \tilde{\boldsymbol{v}}$$

Change due to camera motion

$$\tilde{L}_\tau = \boldsymbol{A}\tilde{\boldsymbol{v}}$$

Render static camera view

$$\tilde{L}(\tau_1) = L(\tau_0) + \tilde{L}_\tau$$

Demo: Change Detection

Estimate camera motion
(closed-form least-squares)

$$\boldsymbol{A}\boldsymbol{v} = L_\tau \rightarrow \tilde{\boldsymbol{v}}$$

Change due to camera motion

$$\tilde{L}_\tau = \boldsymbol{A}\tilde{\boldsymbol{v}}$$

Render static camera view

$$\tilde{L}(\tau_1) = L(\tau_0) + \tilde{L}_\tau$$

Pixel differencing

$$\boldsymbol{R} = L(\tau_1) - \tilde{L}(\tau_1)$$

Demo: Change Detection

Estimate camera motion
(closed-form least-squares)

$$\boldsymbol{A}\boldsymbol{v} = L_\tau \rightarrow \tilde{\boldsymbol{v}}$$

Change due to camera motion

$$\tilde{L}_\tau = \boldsymbol{A}\tilde{\boldsymbol{v}}$$

Render static camera view

$$\tilde{L}(\tau_1) = L(\tau_0) + \tilde{L}_\tau$$

Pixel differencing

$$\boldsymbol{R} = L(\tau_1) - \tilde{L}(\tau_1)$$

Simplifies to plenoptic residual

$$= L_\tau - \tilde{L}_\tau$$

Rejection of Apparent Motion

Input

Rejection of Apparent Motion


Input

Rejection of Apparent Motion

Naive – note apparent motion

Rejection of Apparent Motion

Plenoptic residual

Rejection of Apparent Motion

Input

Rejection of Apparent Motion

Input

Rejection of Apparent Motion

Naive – note apparent motion

Rejection of Apparent Motion

Plenoptic residual

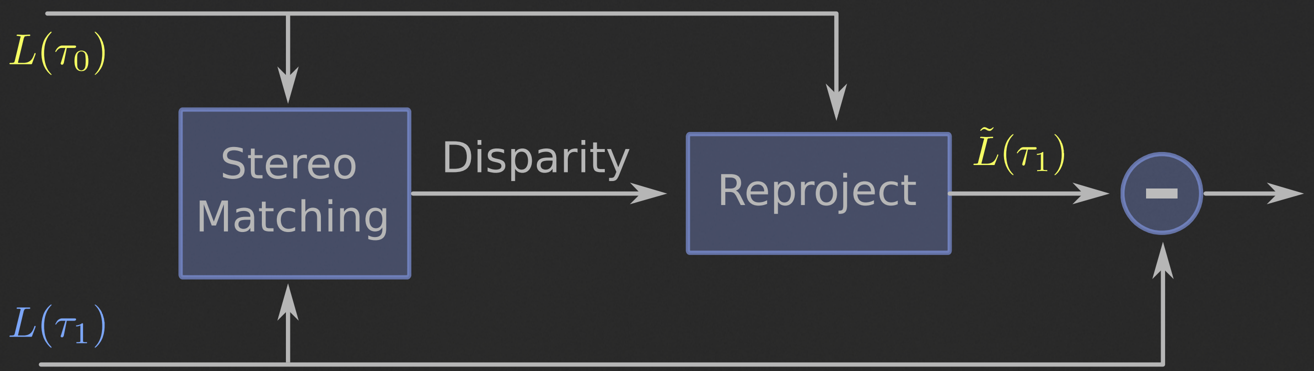
Rejection of Apparent Motion

Scene	L_τ (dB)	R (dB)	Ratio (dB)
Jar	-31.81	-35.848	4.0386
Jar	-27.634	-31.029	3.3954
Jar	-36.452	-43.197	6.7448
Pen	-23.805	-28.842	5.037
Pen	-34.679	-39.917	5.2385
Toothpicks	-33.064	-33.55	0.48605
Toothpicks	-30.576	-32.087	1.5104
Toothpicks	-39.247	-42.276	3.0284
Mean	-29.684	-33.439	4.0905

vs. Structure from Motion

Stereo as stand-in for SfM

- Similar characteristics
- Simplified subset
- Upper bound on performance



vs. Structure from Motion

Input

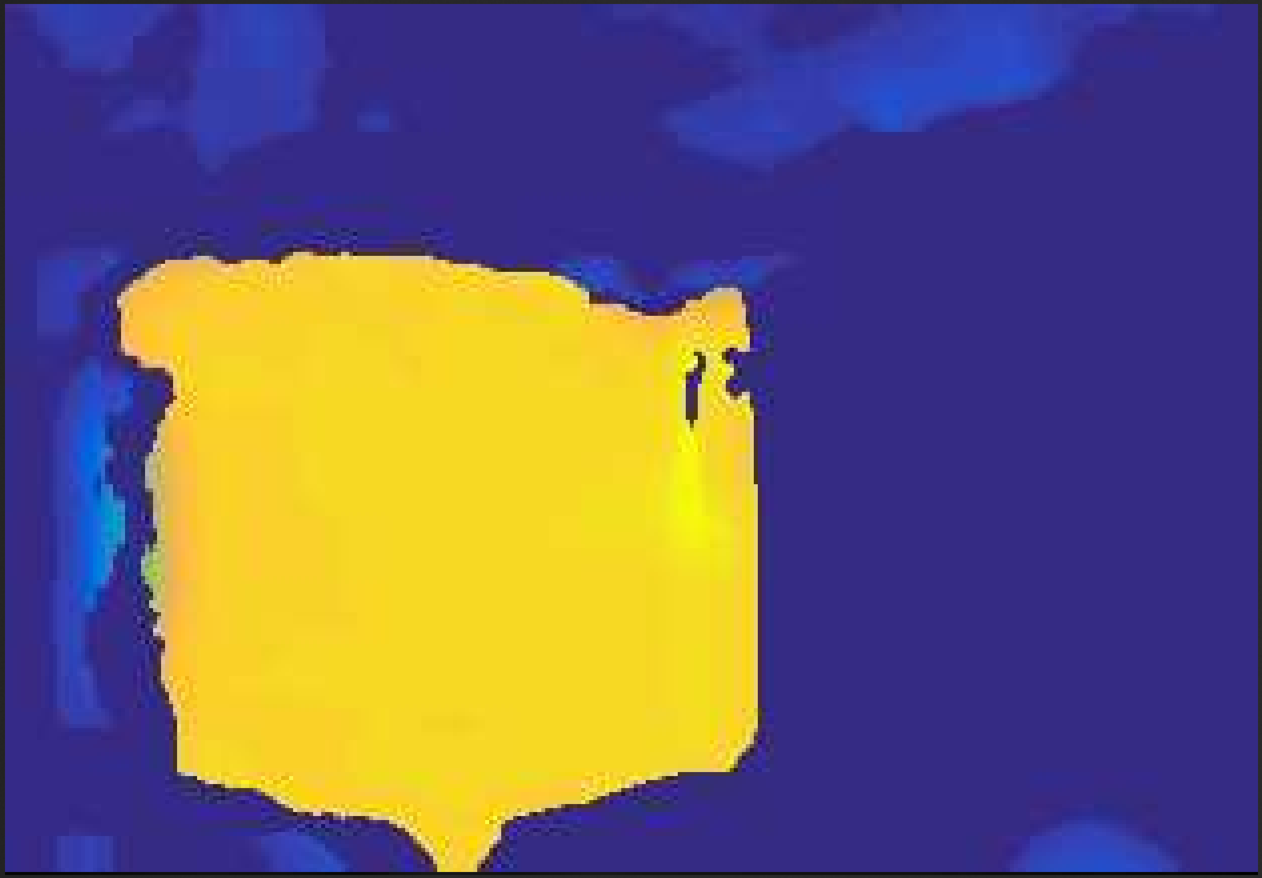
Scene motion aligned with camera motion

vs. Structure from Motion

Input

Scene motion aligned with camera motion

vs. Structure from Motion



Disparity

Scene motion aligned with camera motion

SfM confuses motion for depth

vs. Structure from Motion

Actual change

Scene motion aligned with camera motion

SfM confuses motion for depth

vs. Structure from Motion

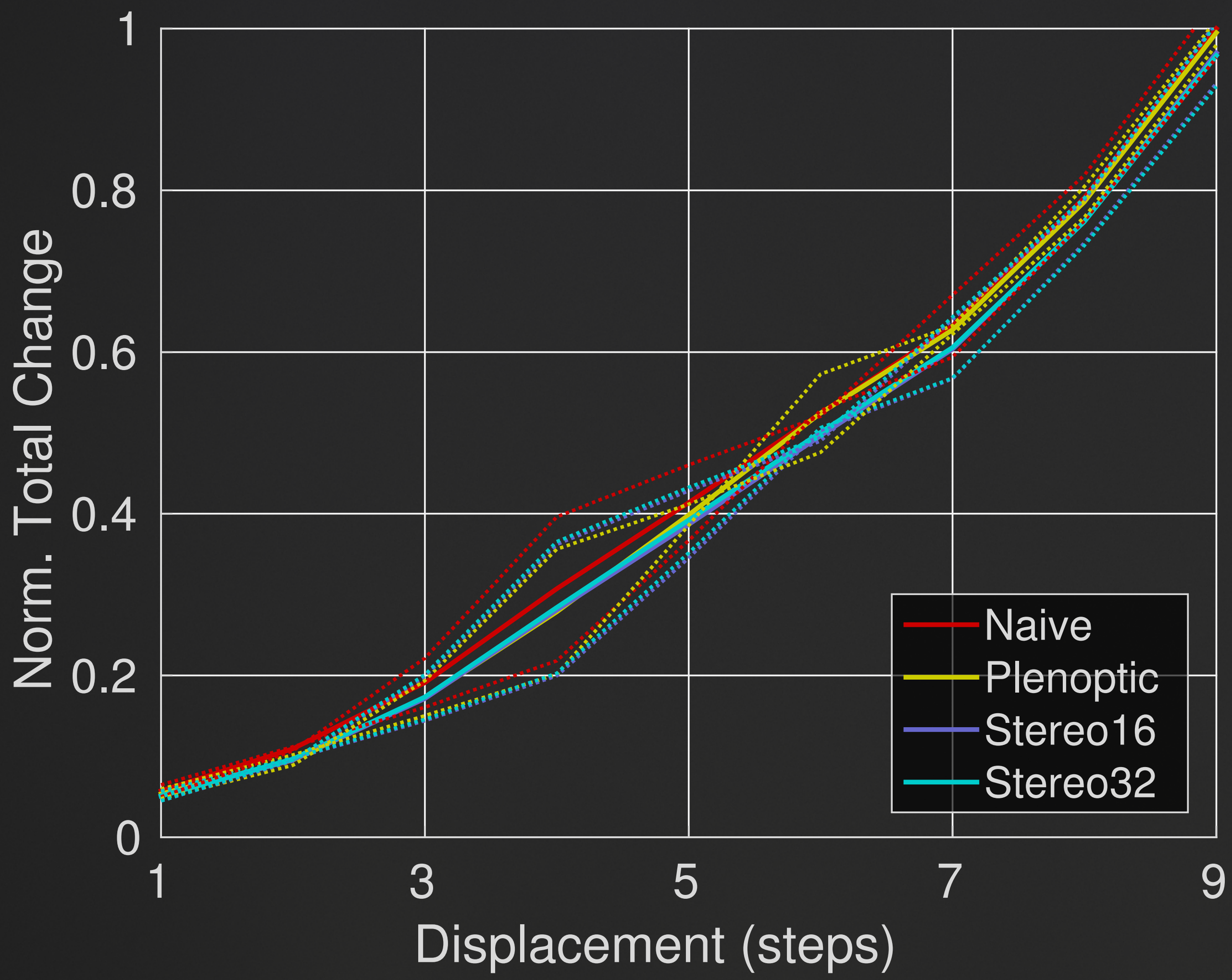
SfM-based change estimate

Scene motion aligned with camera motion

SfM confuses motion for depth

Poor change detection

vs. Structure from Motion



Static camera

Naive = true

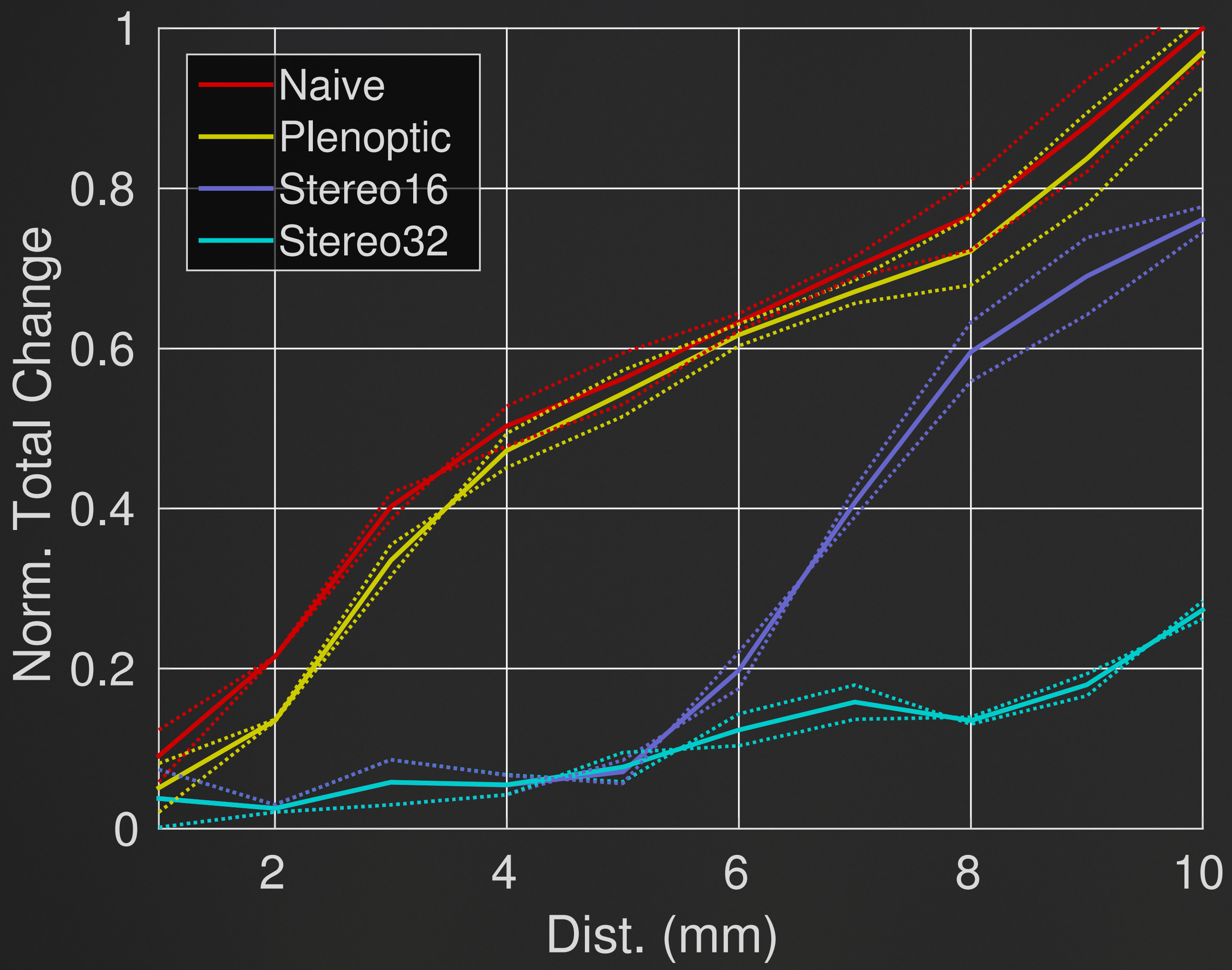
Stereo = stand-in SfM

16/32 = max disparity

Control: vertical motion

All perform well

vs. Structure from Motion

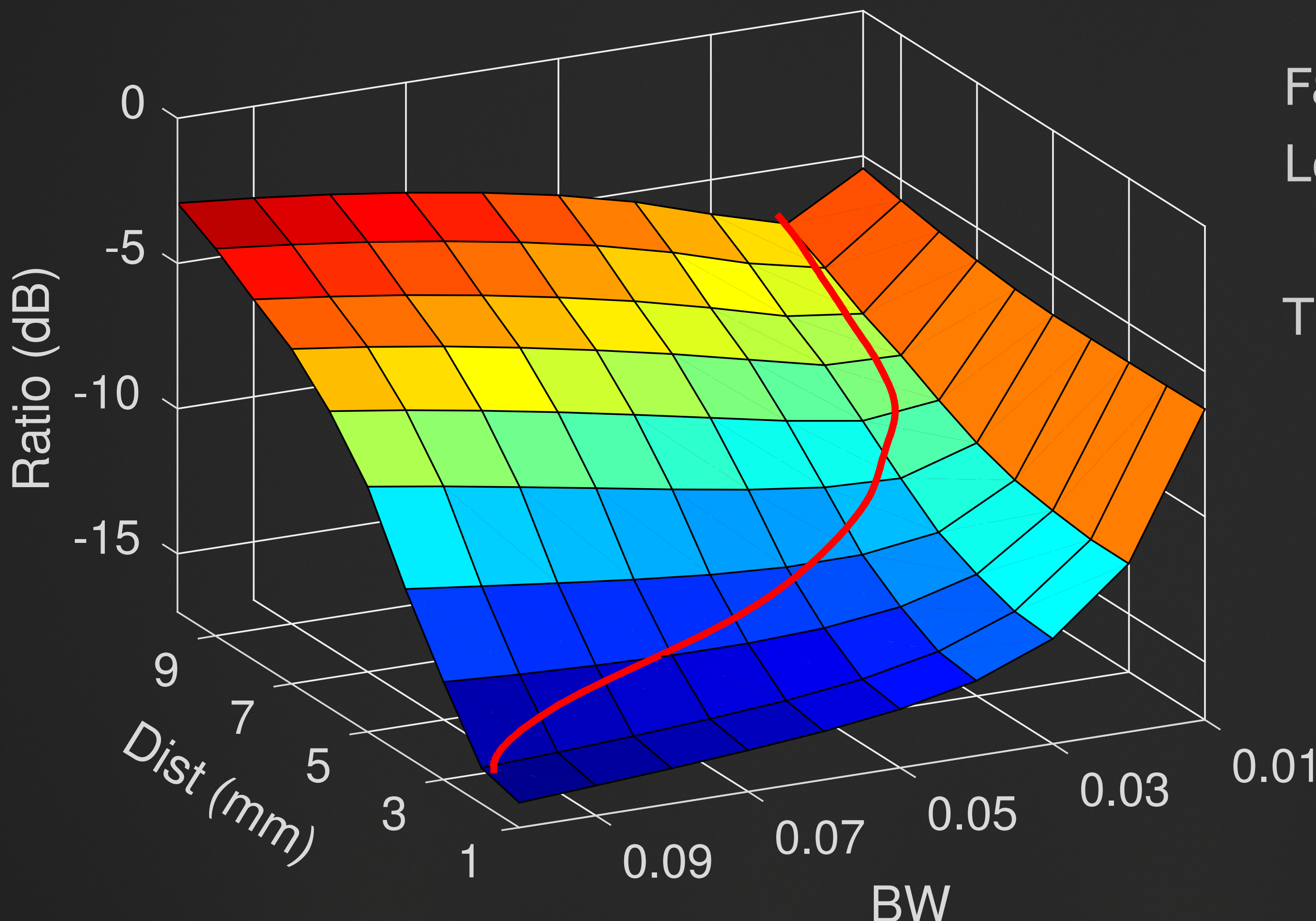


Static camera
Naive = true
Stereo = stand-in SfM
16/32 = max disparity

Translation in x
Poor SfM performance

Maximum Camera Motion

Limitation: small camera motions



False / true positive ratio

Lower = better

Tradeoff:

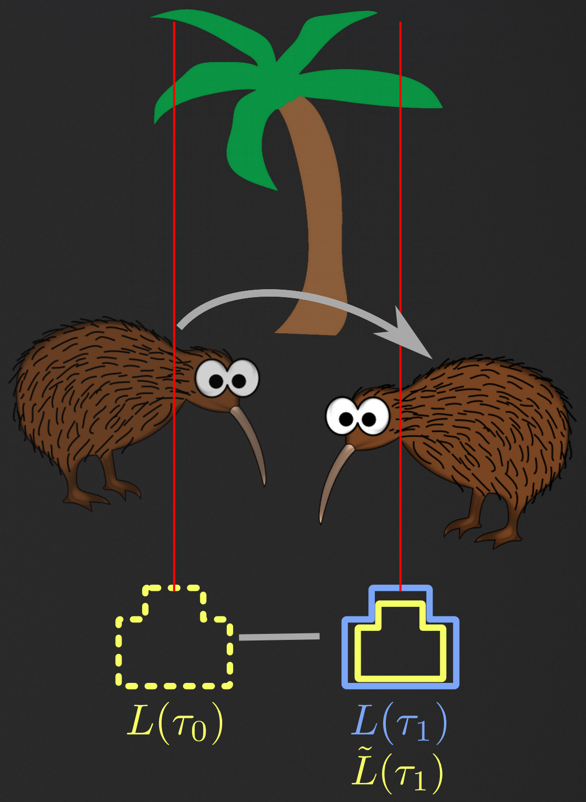
- Sensitivity (BW)
- Max camera motion

Conclusions

- Simplified change detection for moving cameras
  - Closed-form, simple, parallel
  - Outperforms monocular SfM for common scenes
  - Limited camera motion, tradeoff with sensitivity
- Framework to simplify other problems
  - Moving camera → Virtual static camera

What's Next?

- Other still-camera solutions
  - Object tracking, segmentation, isolation and removal, denoising, velocity & temporal filtering
- Better imaging
  - Custom cameras, hybrids
- Simplify difficult tasks
  - 4D algorithms, sensor fusion



Light Field Toolbox for MATLAB

- Load Gantry and Lytro imagery
- Calibrate and rectify Lytro imagery
- Linear depth, volume filters
- Denoising: low-light, fog, dust, murky water
- Occluder removal: rain, snow, silty water

ACRV
AUSTRALIAN CENTRE FOR ROBOTIC VISION

ACFR
AUSTRALIAN CENTRE FOR FIELD ROBOTICS

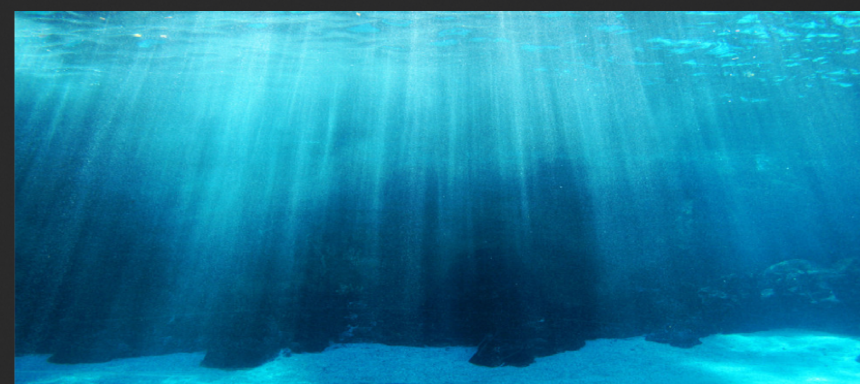
QUT
Queensland University of Technology
Brisbane Australia

THE UNIVERSITY OF SYDNEY

Australian Government
Australian Research Council

Challenges in Robotic Vision

Environment



- Variable light: Day & night
- Weather
- Participating media
- Unstructured, dynamic scenes

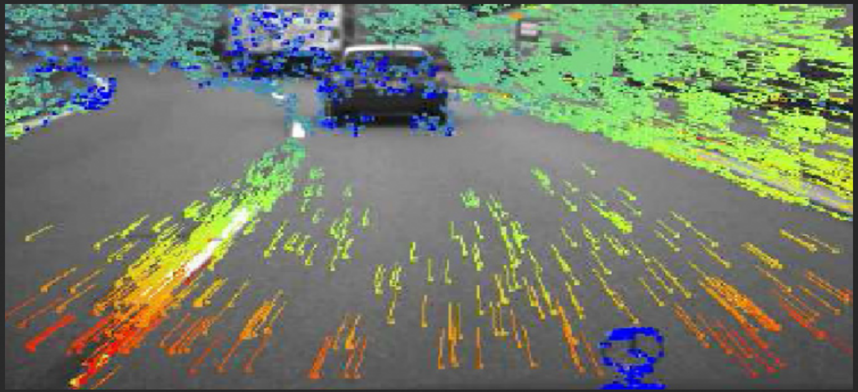
Challenges in Robotic Vision

Platform

- Power, computing
- Weight, volume
- Actuation
- Time

Challenges in Robotic Vision

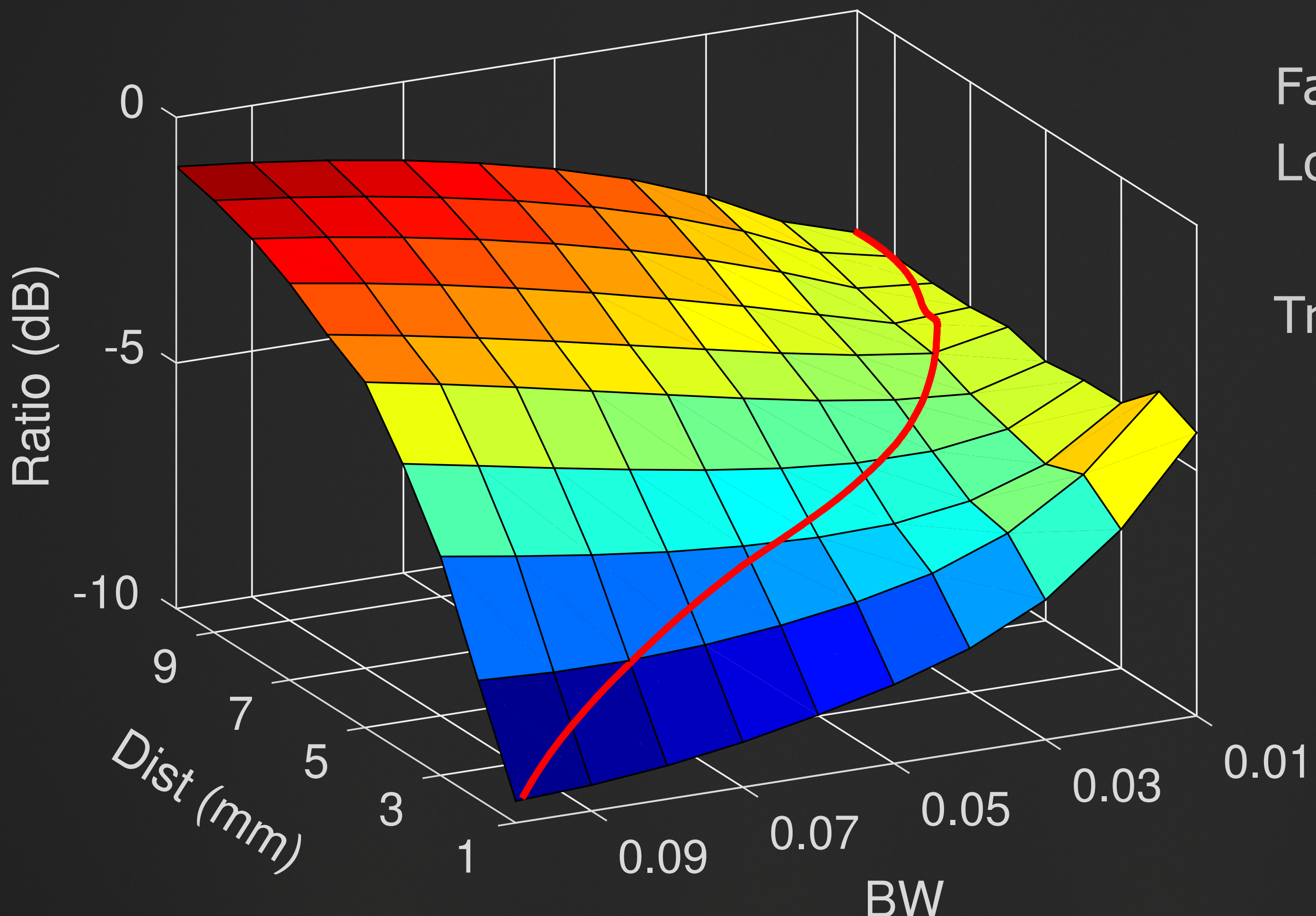
Camera



- Light vs. depth of field
- Light vs. motion blur
- Nonunifom apparent motion

Maximum Camera Motion

Limitation: small camera motions, static scene



False / true positive ratio

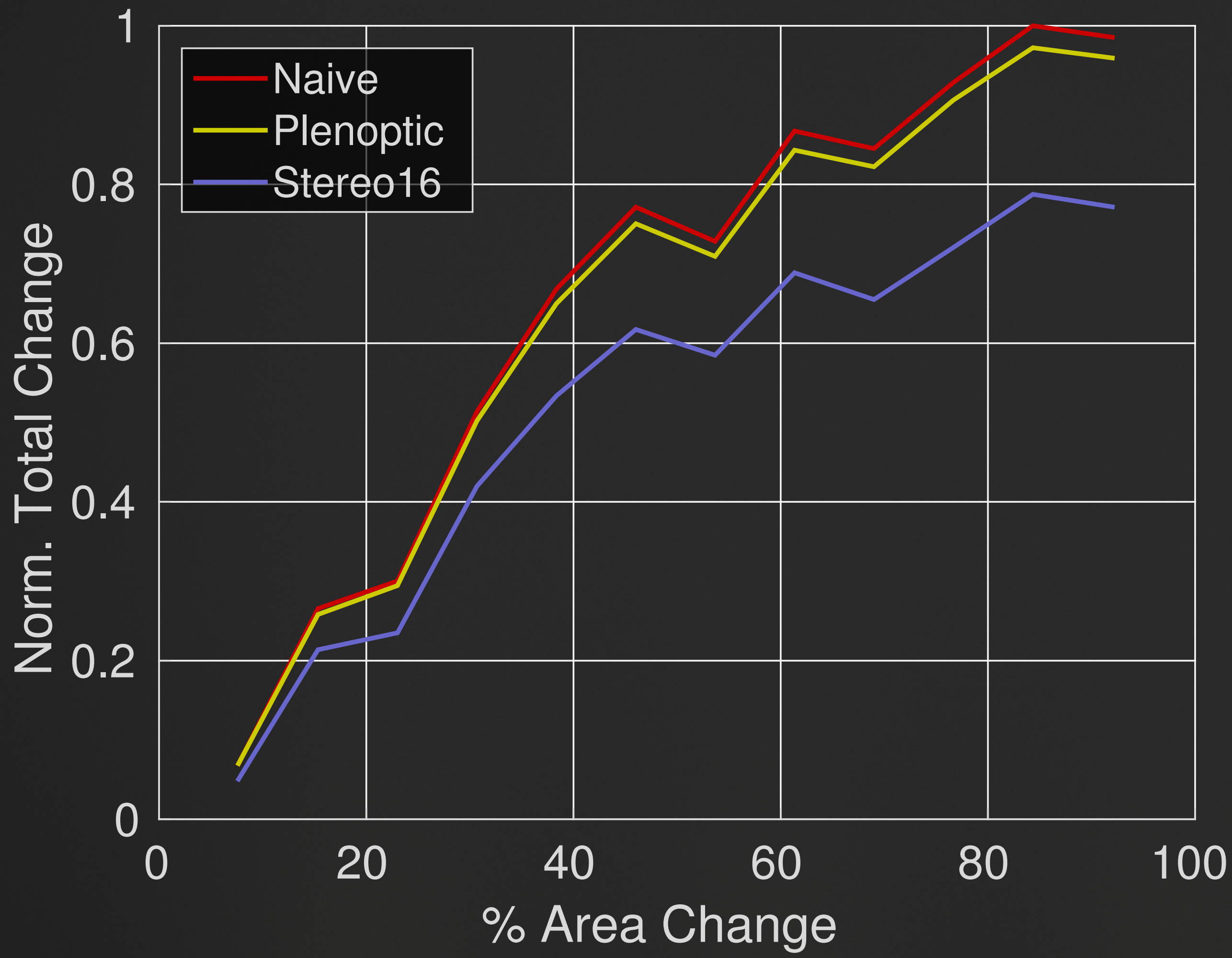
Lower = better

Tradeoff:

- Sensitivity (BW)
- Max camera motion

Maximum Scene Motion

Limitation: what if the whole scene is moving?



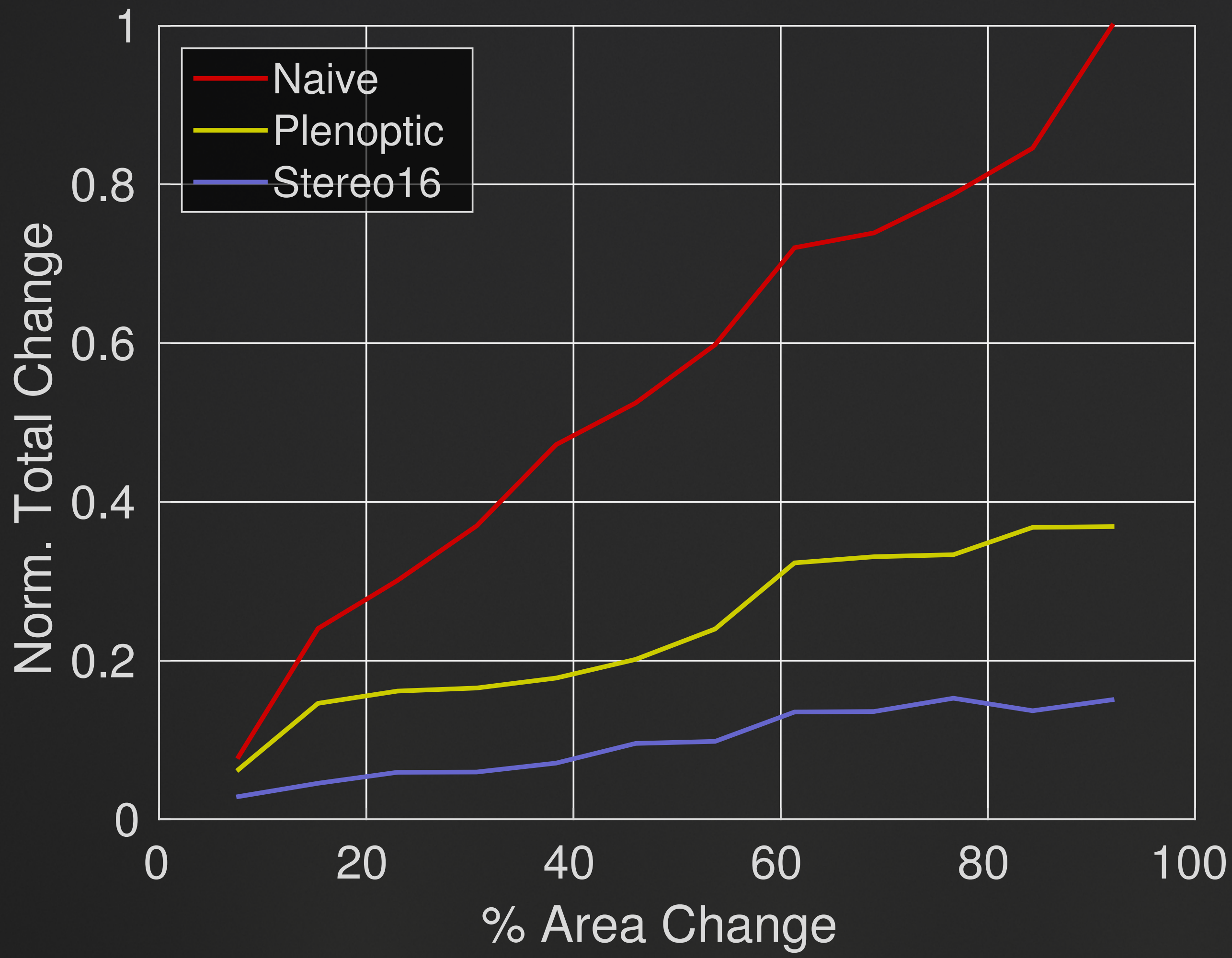
Static camera

Naive = true

- Small motions
- Random, incoherent
- Good performance

Maximum Scene Motion

Limitation: what if the whole scene is moving?



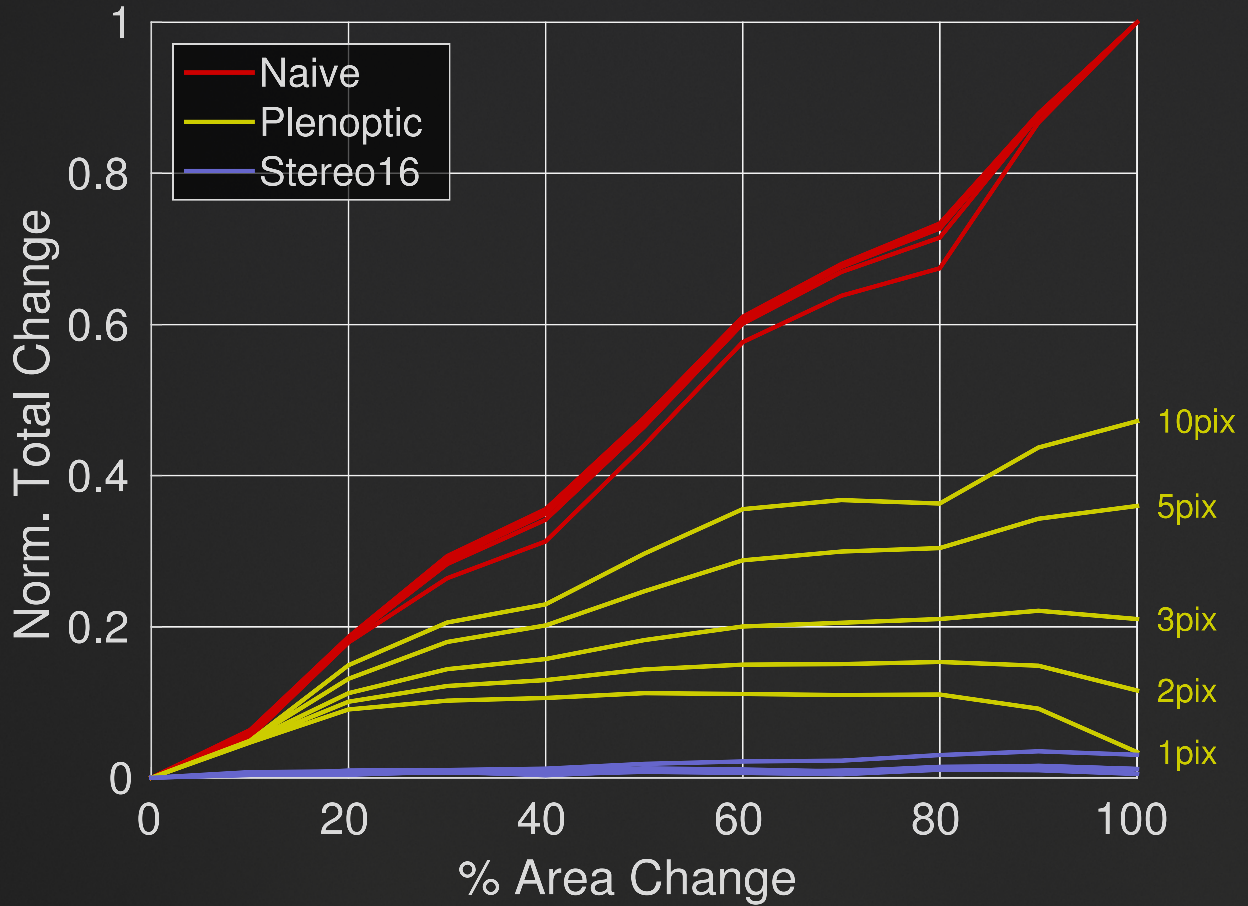
Static camera

Naive = true

- Small motions
- Coherent, appears as apparent motion
- Poor performance

Maximum Scene Motion

Limitation: what if the whole scene is moving?



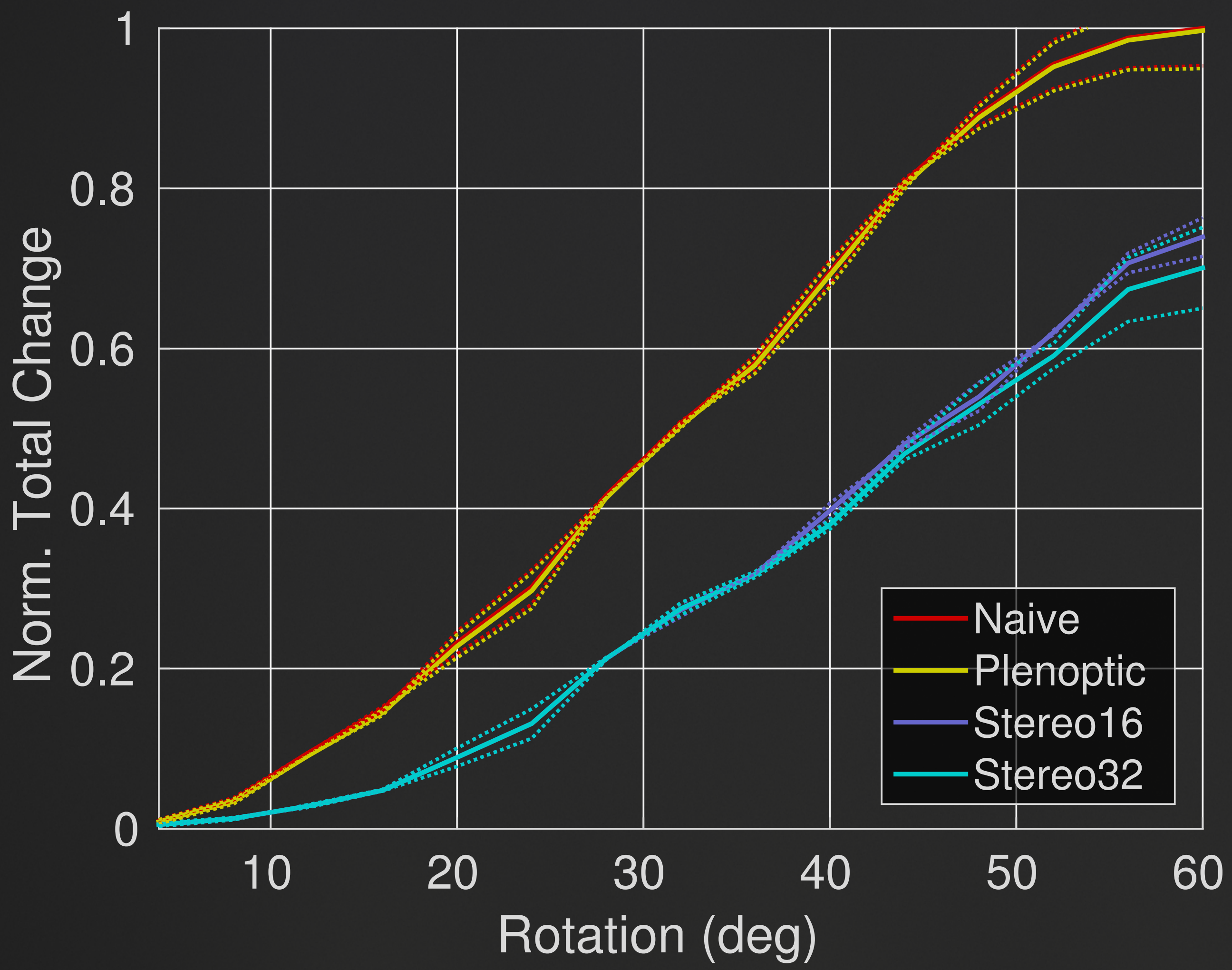
Static camera

Naive = true

Simulation to find worst performance:

- Small motion
- Across whole image
- Coherent, appears as apparent motion

vs. Structure from Motion



Static camera

Naive = true

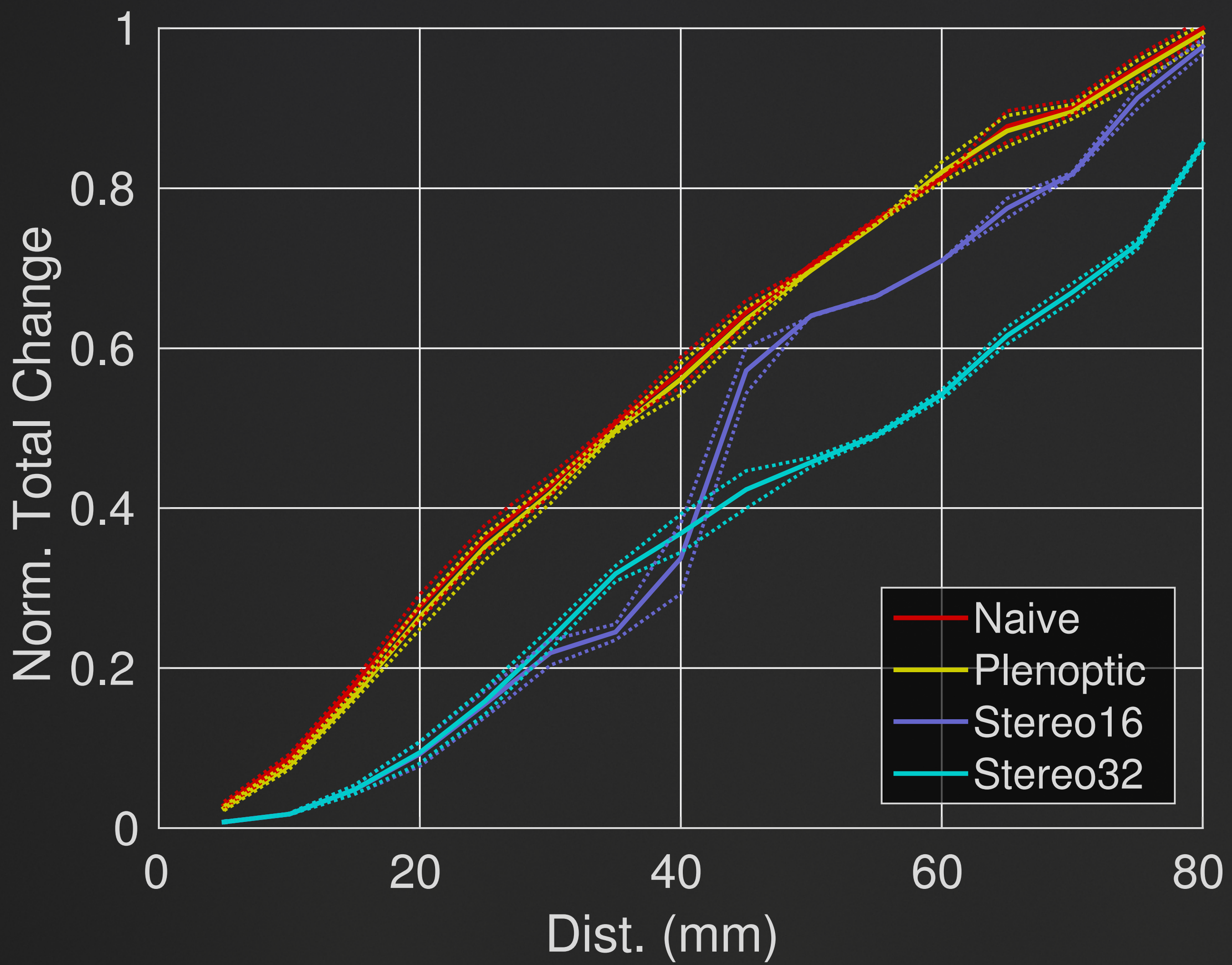
Stereo = stand-in SfM

16/32 = max disparity

Rotation about vertical

Poor SfM performance

vs. Structure from Motion



Static camera

Naive = true

Stereo = stand-in SfM

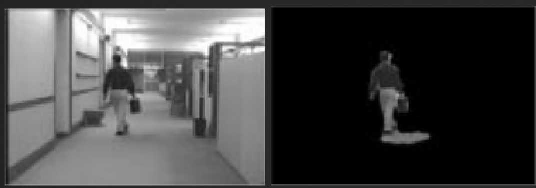
16/32 = max disparity

Translation in x,z

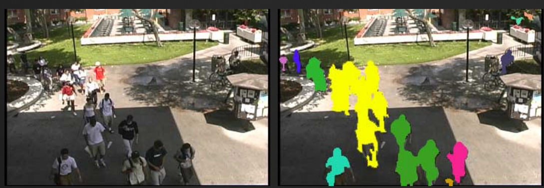
Poor SfM performance

Stationary Cameras Simplify Things

Denoising [Bennett2005]



Change Detection [Chien2002]



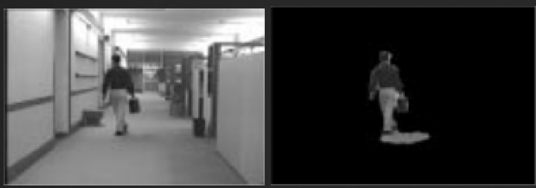
Tracking [Zhao2002]

Classic, simple video processing

- Denoising
- Change detection
- Tracking
- Segmentation
- Temporal filtering

Stationary Cameras Simplify Things

Denoising [Bennett2005]

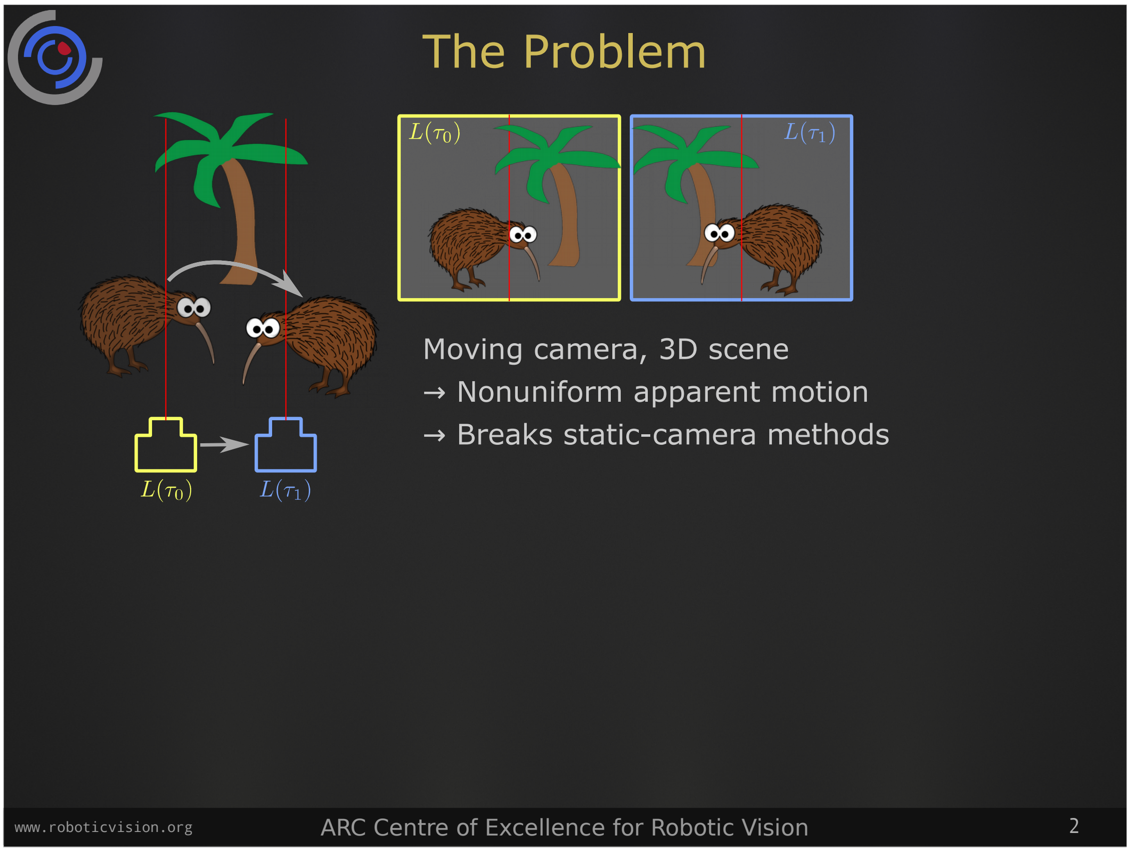


Change Detection [Chien2002]

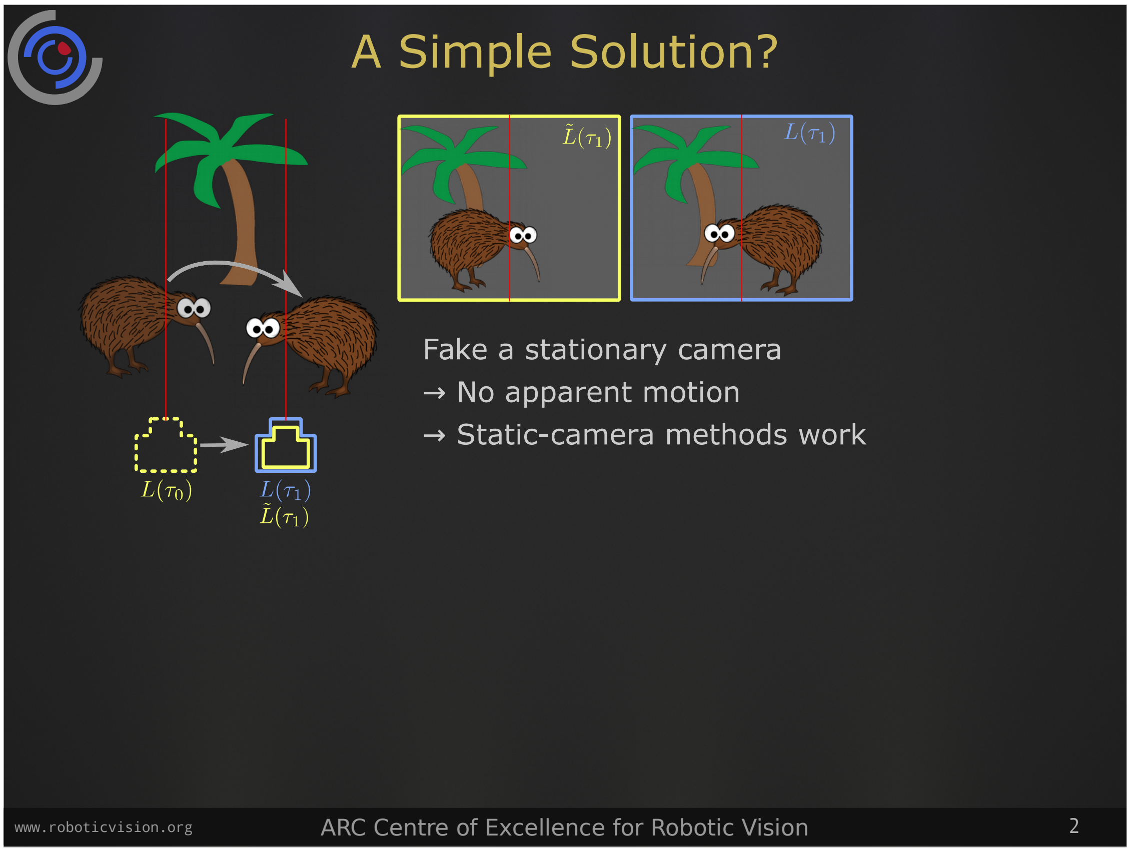
Tracking [Zhao2002]

Classic, simple video processing
- Denoising
- Change detection
- Tracking
- Segmentation
- Temporal filtering

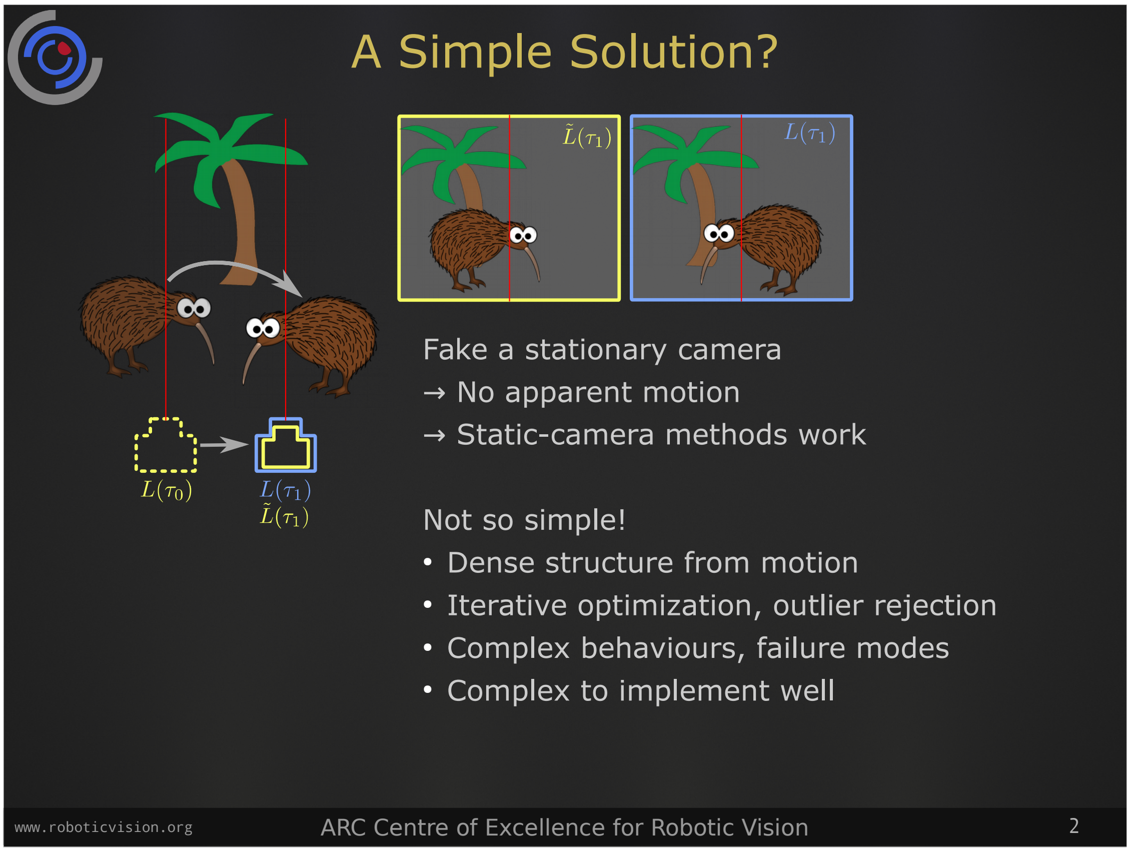
But these break
when the camera moves



The camera moves between t0, t1
Causes apparent motion, e.g. the tree
This breaks simple methods
e.g. makes it hard to distinguish genuine motion from apparent motion



Render a virtual stationary camera
The camera at t0 is replaced with a virtual camera aligned with the one at t1

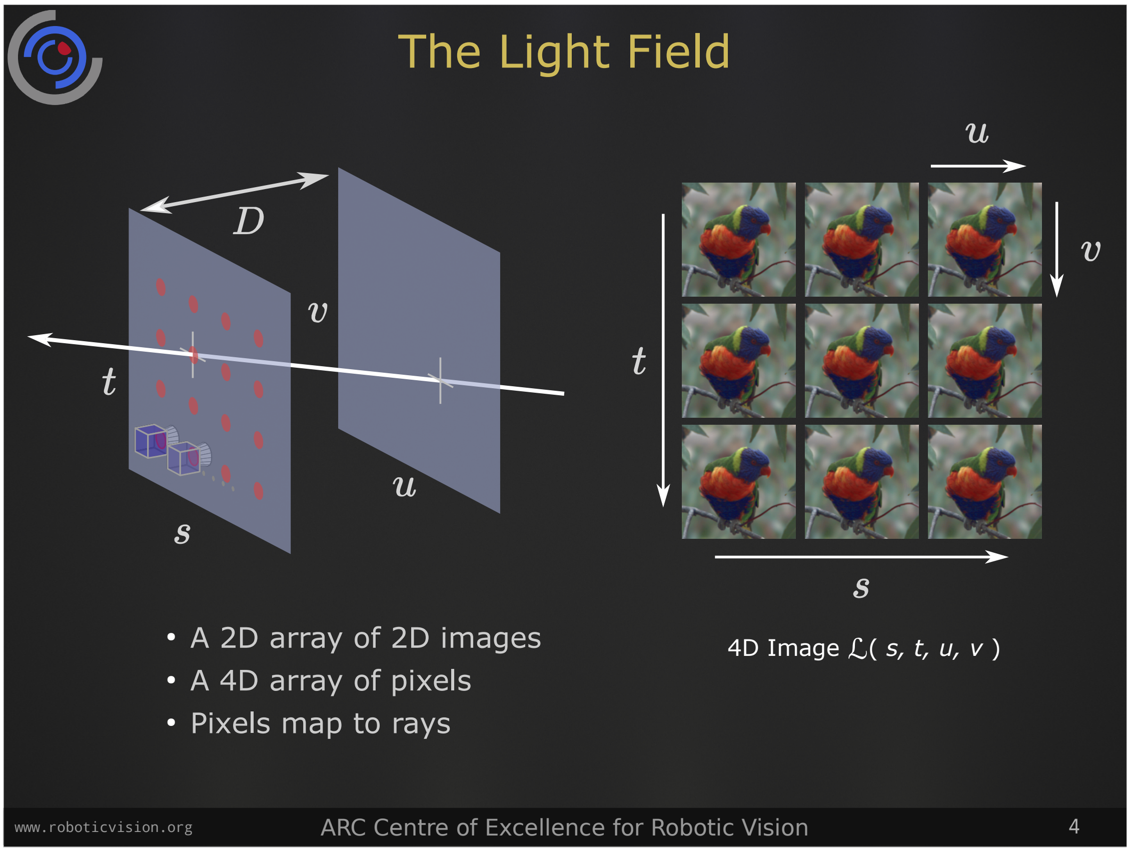


Setting up a need for LF cams here

All LF cams measure the same form of information as the big array on the left

More practical form factors are increasingly available

Pelican imaging and Linx are italicized because you can't buy them yet



A camera array measures an array of images

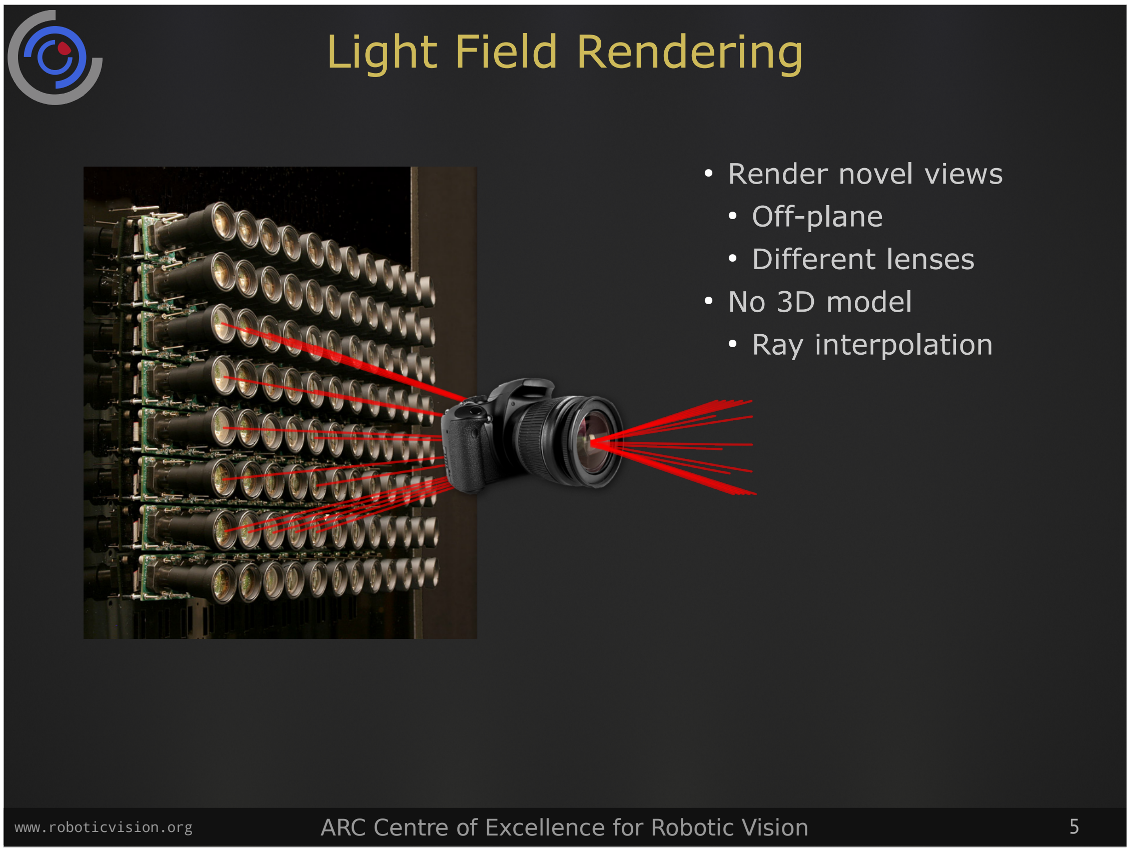
To pick a pixel from this 2D array of 2D images we need 4 numbers:

2 for the camera (s,t)

2 for the pixel (u,v)

This can be seen as a 4D array of pixels

Each pixel maps to a unique ray in the scene, following the two-plane parameterization on the left

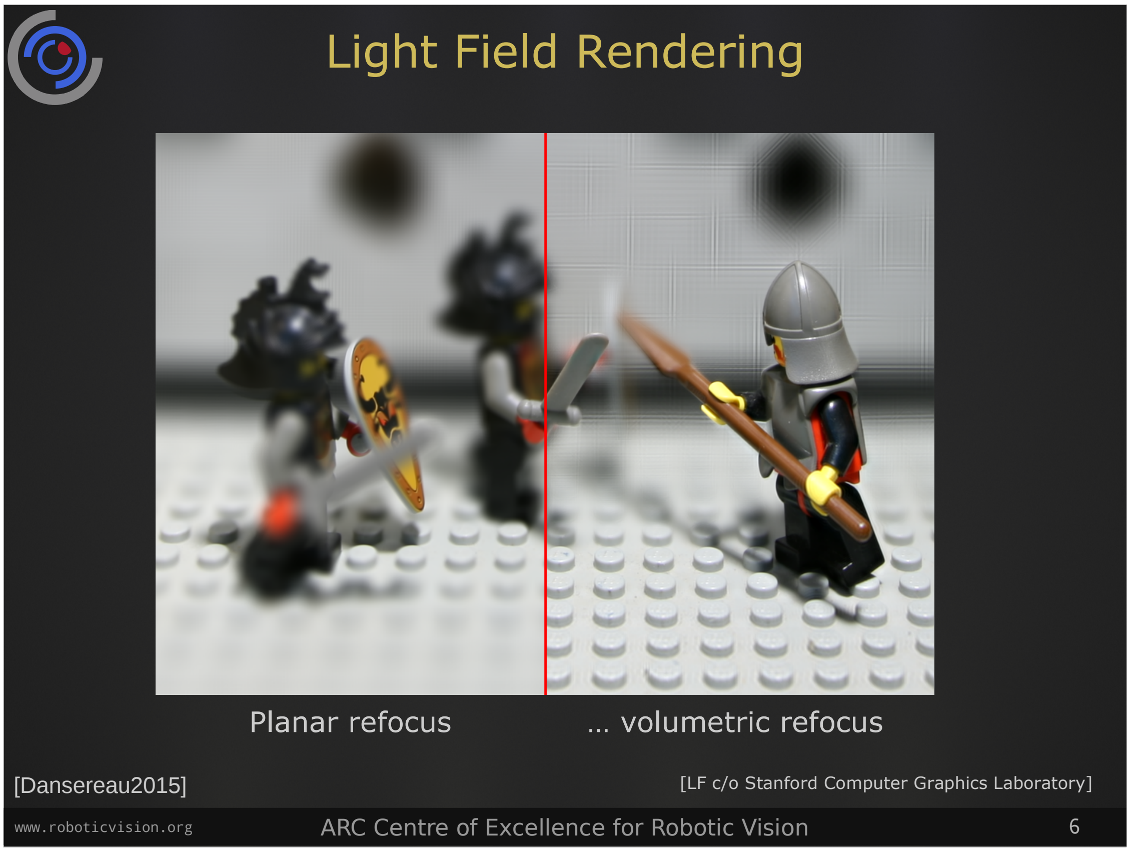


The next few slides should go quickly…

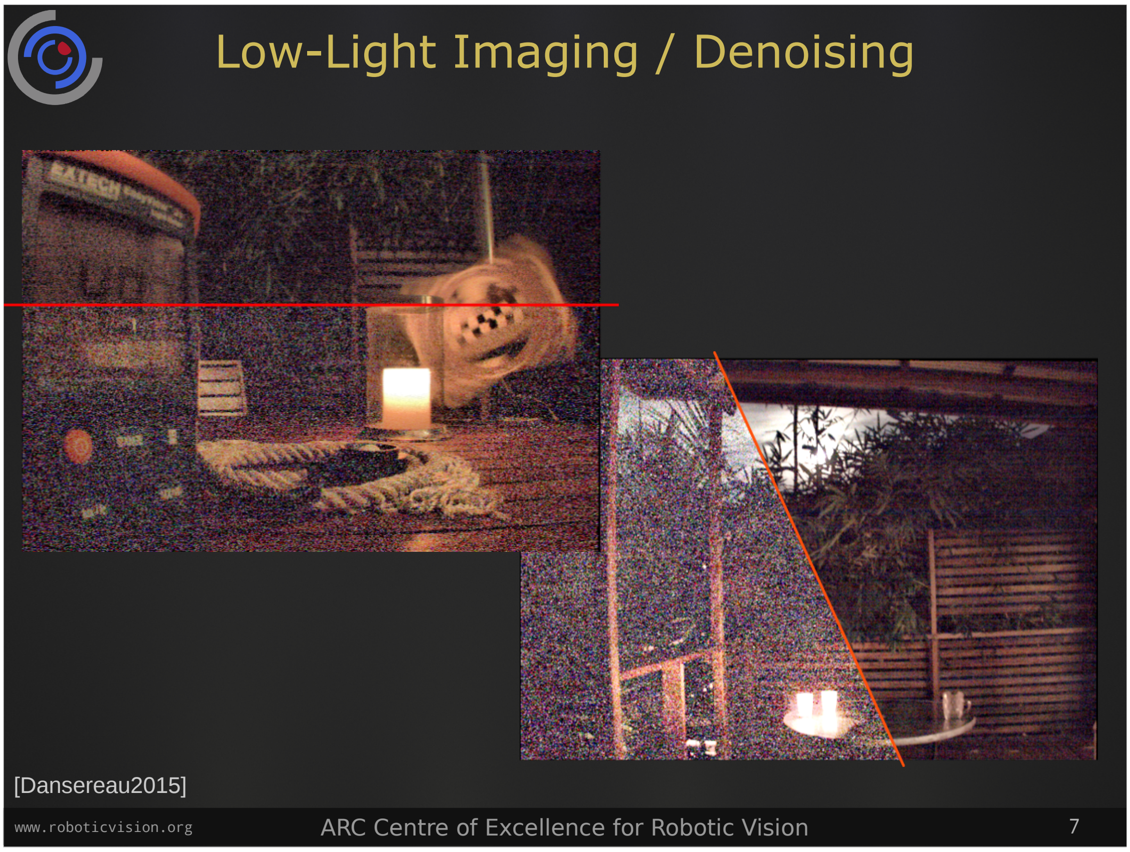
Rendering novel views is simple, with no explicit modelling required

Views can be off the plane of the camera

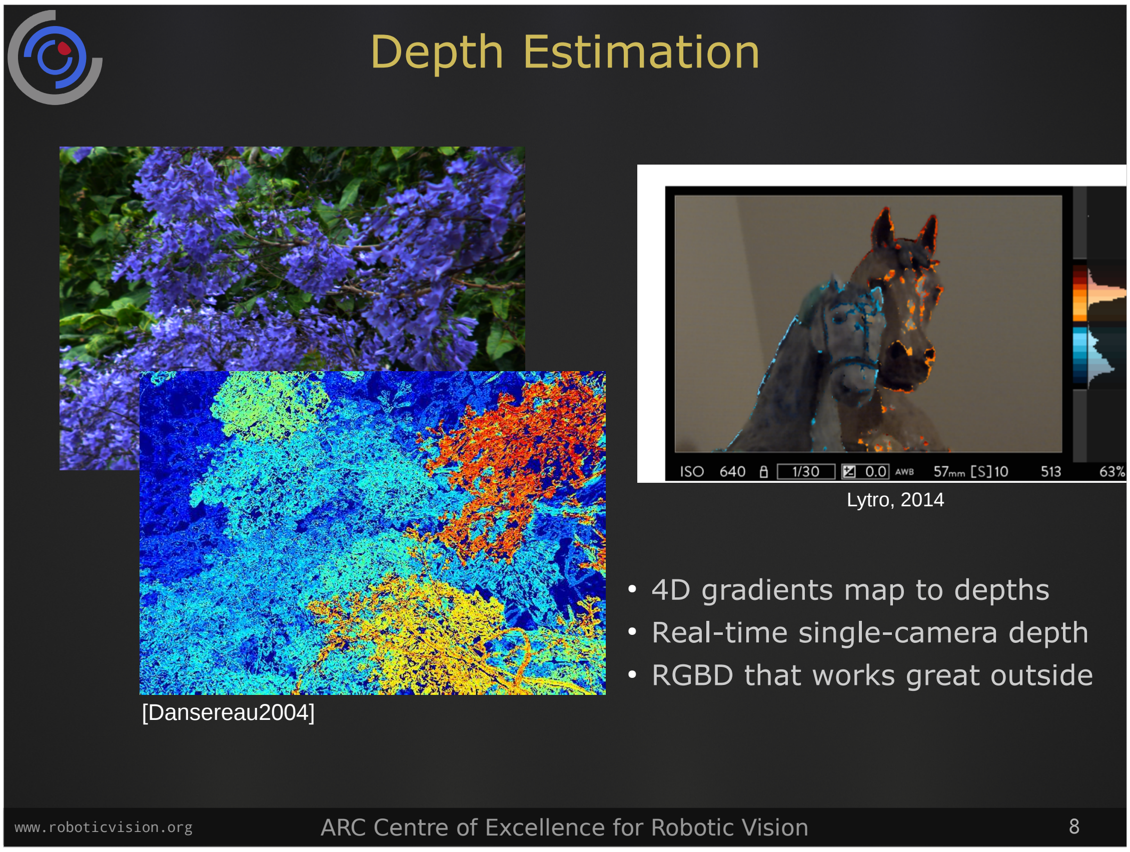
Views can have different optical properties from the camera that measured the light field, e.g. refocus



An example of refocus, including a type not easily achieved with normal cameras



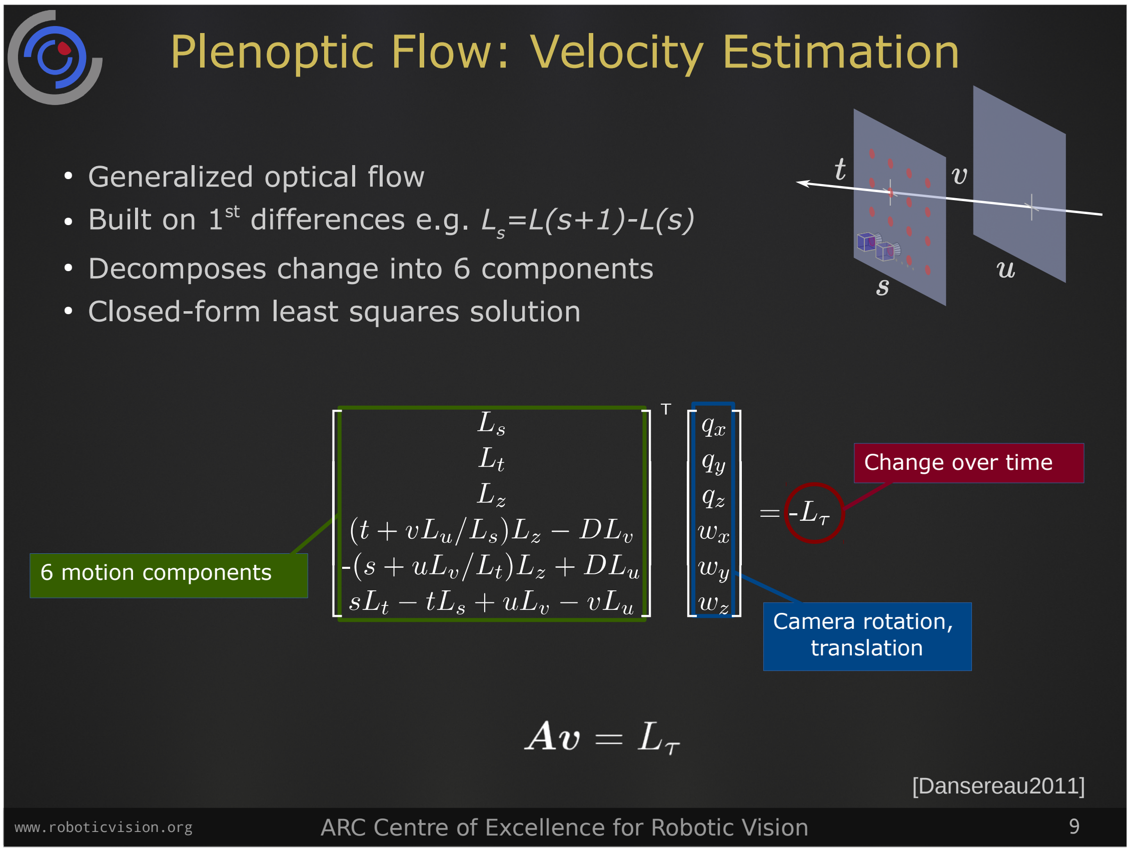
Very fast application: denoising / imaging in low light or through fog or murky water



Fast app: depth information is implicitly captured by the light field

Depth estimation can be simple, real-time

Currently runs in real-time on Lytro Illum cameras



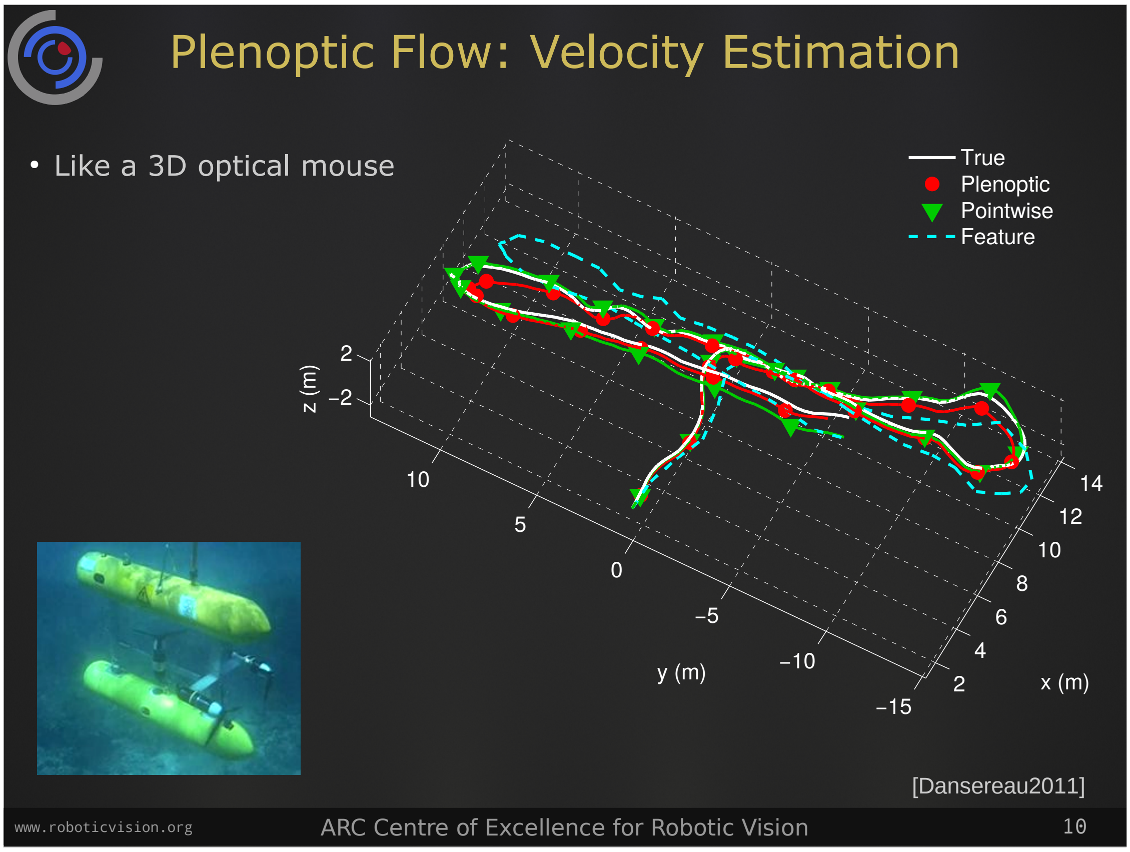
Plenoptic flow in detail

Point out the high level (coloured boxes) and that there are 6 components that we can visualize (next slide)

Point out that the whole this is a classic overdetermined linear system of equations

The 1st differences are a bit of a cheat, there's an extra step not shown, to get from the sample LF to continuous-space quantities

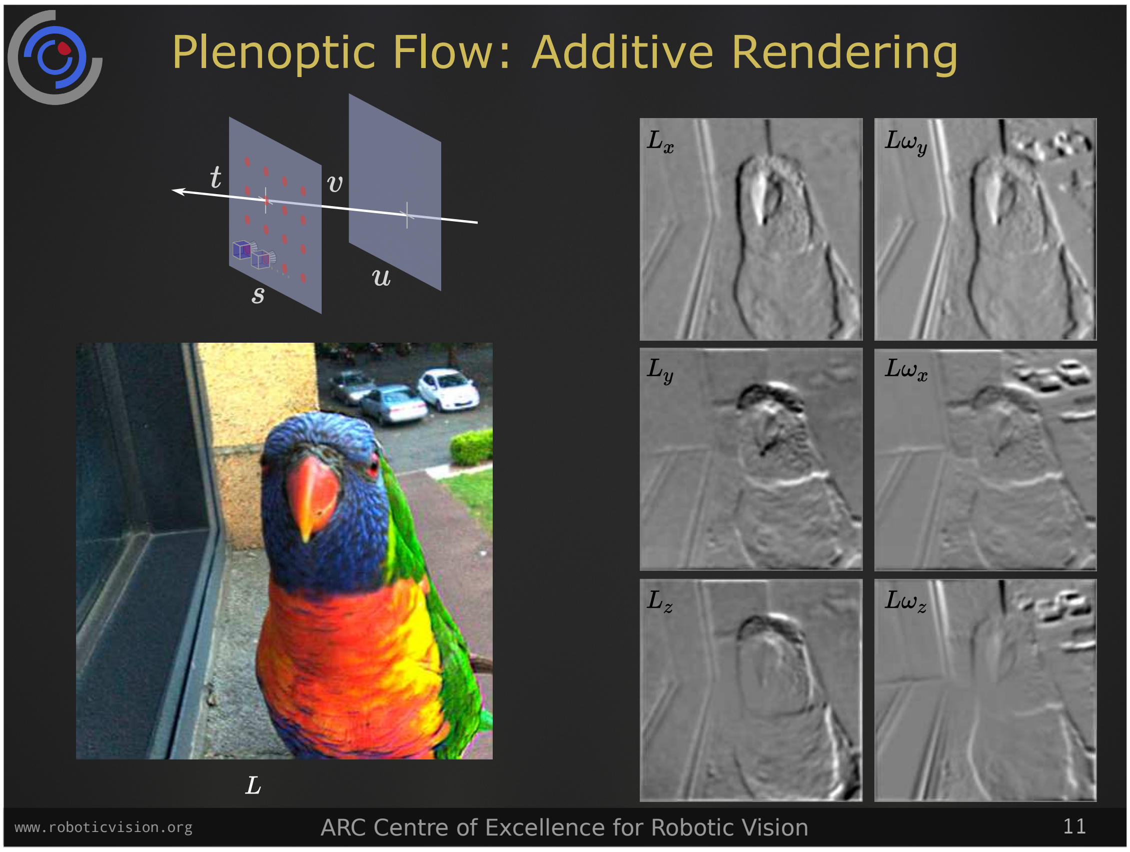
Closed-form least squares solution



Closed-form visual odometry isn't usually possible

Result shown is rendered video from real AUV trajectory

Shows plenoptic keeping up with, indeed outperforming leading features + RANSAC method of the time

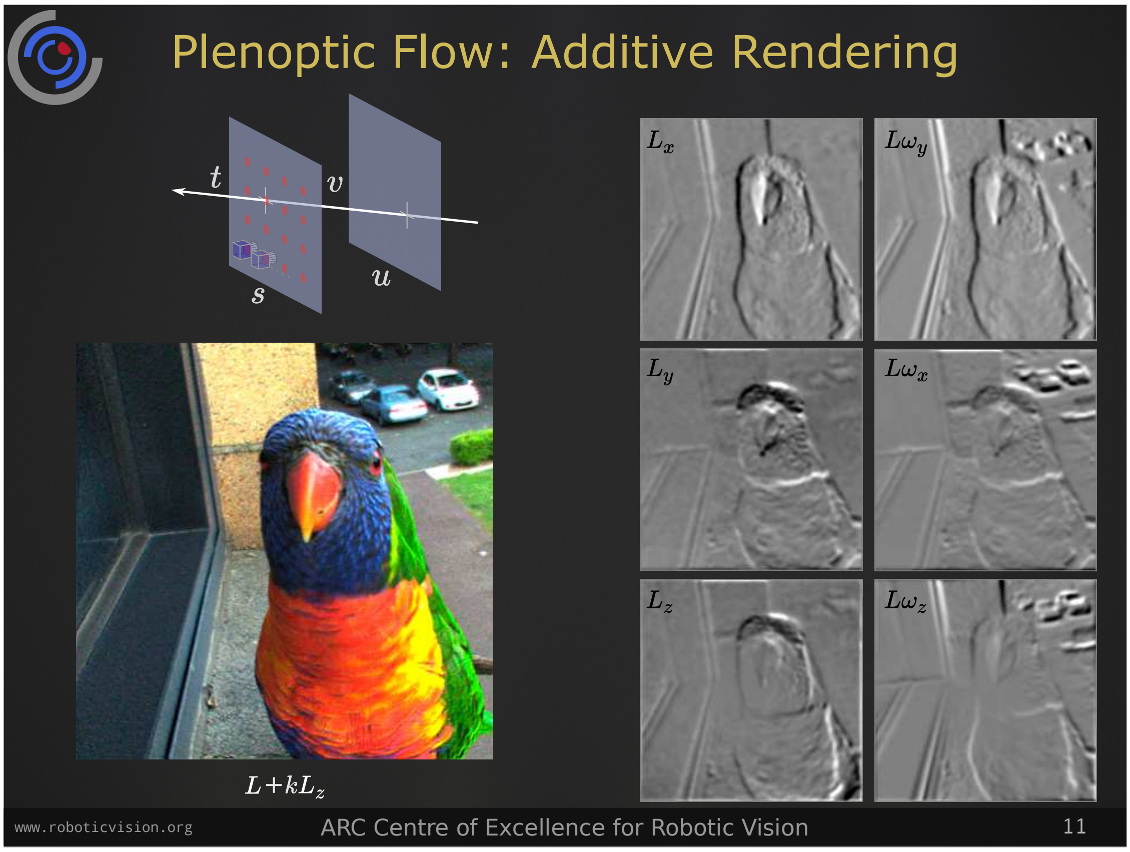


Visualizing the 6 components from the previous slide;

Lx for example is the result of moving the camera to the right

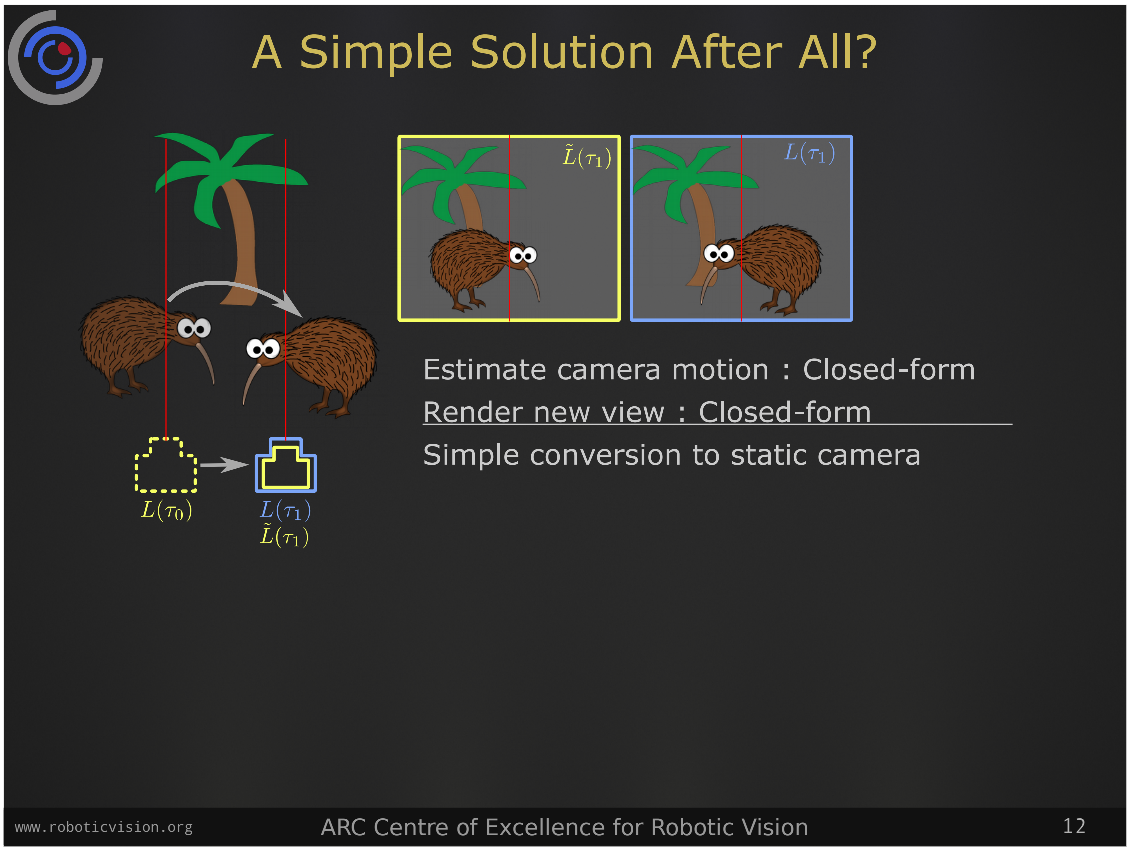
Lwy is rotation about y

To test if these are correct we can add them back to the light field



So this is a very simple way of rendering under small changes

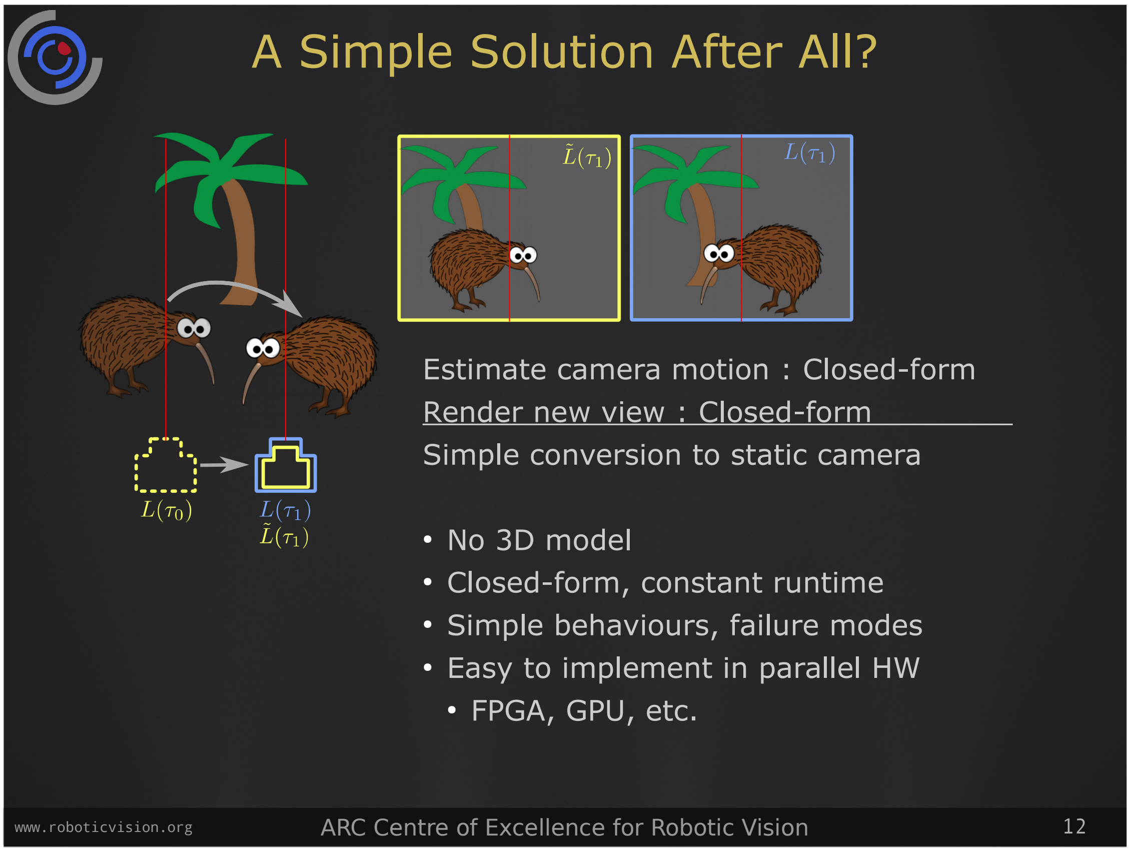
We'll use this to further simplify change detection in the coming example



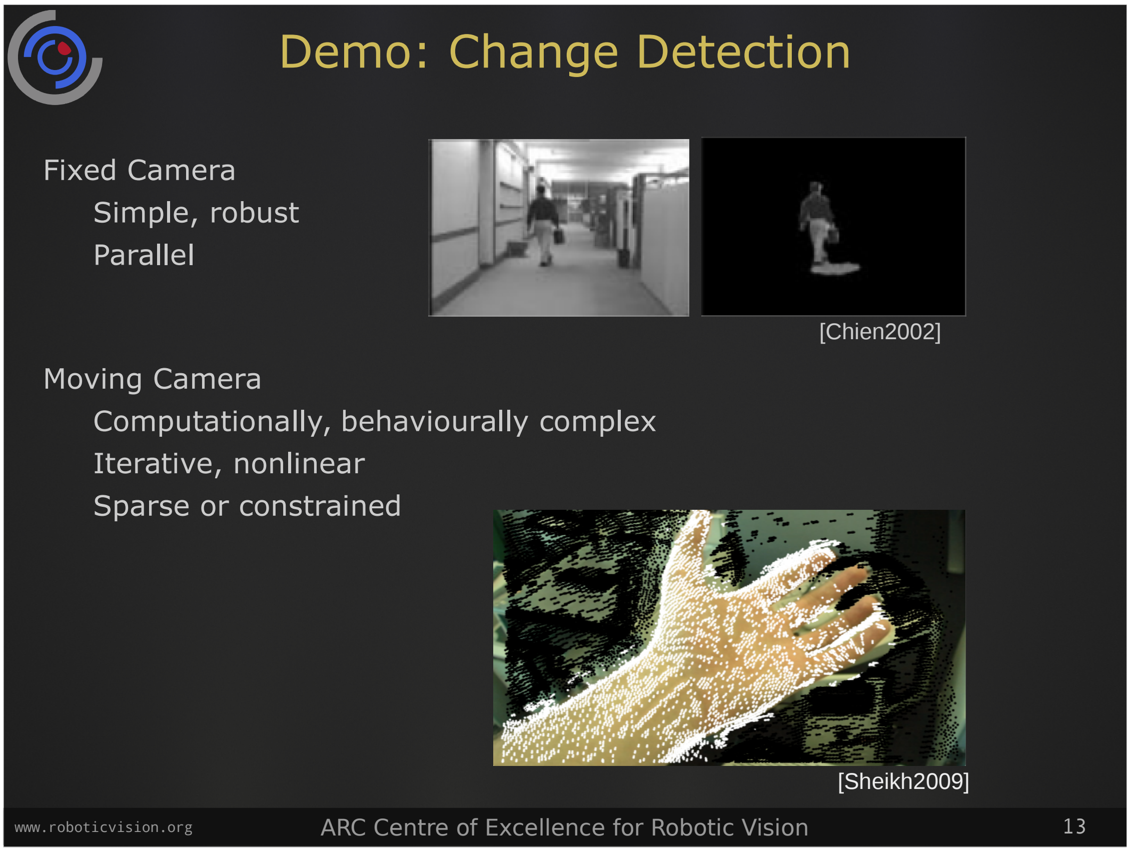
Perhaps with this bag of tricks, LF cams can make our solution a simple one after all

Visual odometry via plenoptic flow is closed-form

Rendering is simple, should be achievable very fast, even closed-form



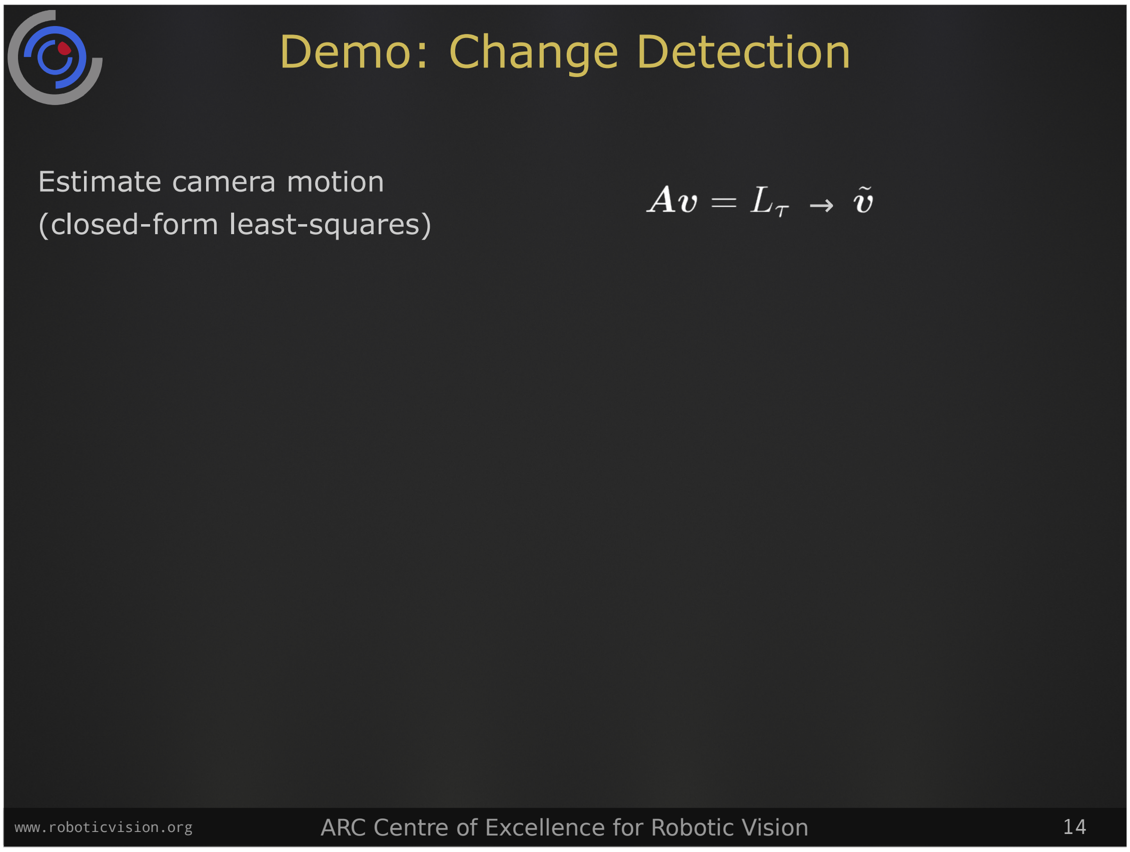
Contrasting against the earlier setup of complex SfM solutions



Let's show off the method with a specific example

Again contrasting the simple, still-camera method with what's necessary using current state-of-the-art techniques for change detection

Sparse as in not all pixels are assigned estimates



Math time
It should go fast

We estimate the camera's velocity using plenoptic flow

Demo: Change Detection

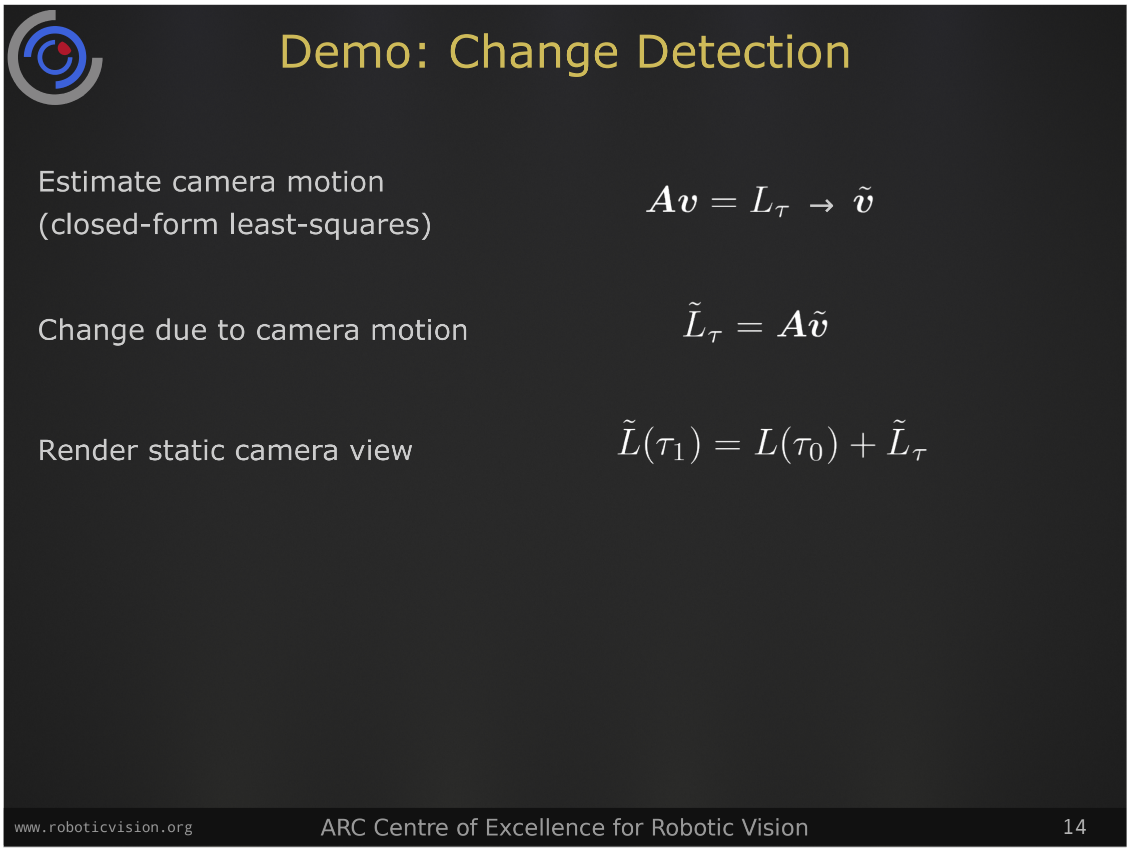
Estimate camera motion
(closed-form least-squares)

$$\boldsymbol{A}\boldsymbol{v} = L_\tau \rightarrow \tilde{\boldsymbol{v}}$$

Change due to camera motion

$$\tilde{L}_\tau = \boldsymbol{A}\tilde{\boldsymbol{v}}$$

This velocity yields an estimated change due to the camera's motion



Adding this estimated change to the LF yields a novel view

This is the still camera we were looking for

Demo: Change Detection

Estimate camera motion (closed-form least-squares) $\boldsymbol{A}\boldsymbol{v} = L_\tau \rightarrow \tilde{\boldsymbol{v}}$

Change due to camera motion $\tilde{L}_\tau = \boldsymbol{A}\tilde{\boldsymbol{v}}$

Render static camera view $\tilde{L}(\tau_1) = L(\tau_0) + \tilde{L}_\tau$

Pixel differencing $\boldsymbol{R} = L(\tau_1) - \tilde{L}(\tau_1)$

This is the first change detection-specific step
Simple pixel differencing

Demo: Change Detection

Estimate camera motion (closed-form least-squares) $\boldsymbol{A}\boldsymbol{v} = L_\tau \rightarrow \tilde{\boldsymbol{v}}$

Change due to camera motion $\tilde{L}_\tau = \boldsymbol{A}\tilde{\boldsymbol{v}}$

Render static camera view $\tilde{L}(\tau_1) = L(\tau_0) + \tilde{L}_\tau$

Pixel differencing $\boldsymbol{R} = L(\tau_1) - \tilde{L}(\tau_1)$

Simplifies to plenoptic residual $= L_\tau - \tilde{L}_\tau$

Remember Lt=L(t1)-L(t0)

Rejection of Apparent Motion

Input

Results, should go fast

Rejection of Apparent Motion

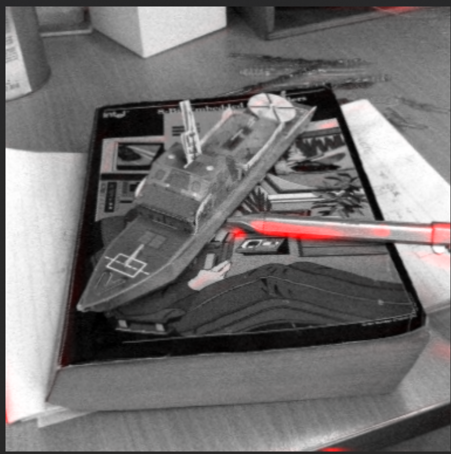
Input

Flip back and forth a few times

Rejection of Apparent Motion

Naive – note apparent motion

Rejection of Apparent Motion



Plenoptic residual

Flipping is helpful

Rejection of Apparent Motion

Input

Rejection of Apparent Motion

Input

Rejection of Apparent Motion

Naive – note apparent motion

Rejection of Apparent Motion

Plenoptic residual

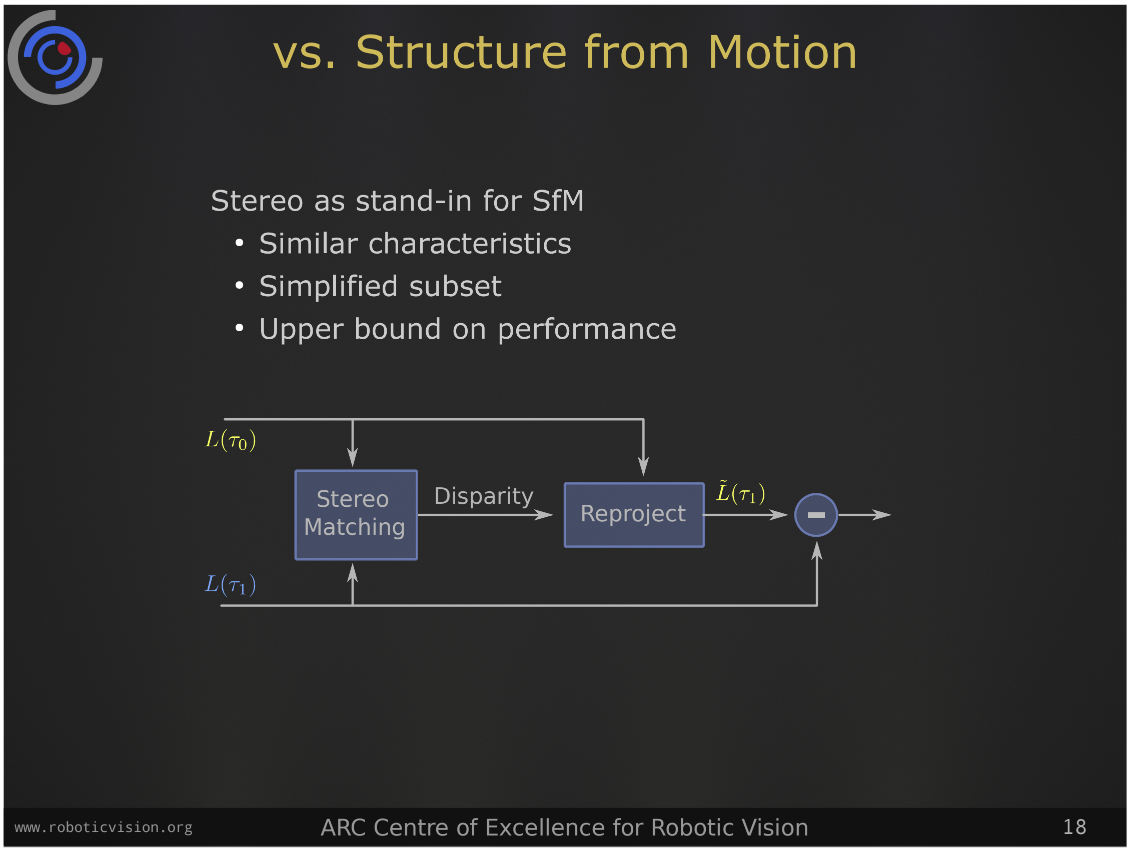
Rejection of Apparent Motion

Scene	L_τ (dB)	R (dB)	Ratio (dB)
Jar	-31.81	-35.848	4.0386
Jar	-27.634	-31.029	3.3954
Jar	-36.452	-43.197	6.7448
Pen	-23.805	-28.842	5.037
Pen	-34.679	-39.917	5.2385
Toothpicks	-33.064	-33.55	0.48605
Toothpicks	-30.576	-32.087	1.5104
Toothpicks	-39.247	-42.276	3.0284
Mean	-29.684	-33.439	4.0905

Comparing naive pixel differencing Lt to the proposed method

Lt is susceptible to apparent motion

The mean ratio of 4 shows our method rejects apparent motion



This isn't mentioned in the abstract
Let's compare with an SfM approach

SfM is used to align two views, and the error taken to effect change detection

What if the motion is parallel with camera motion?

vs. Structure from Motion

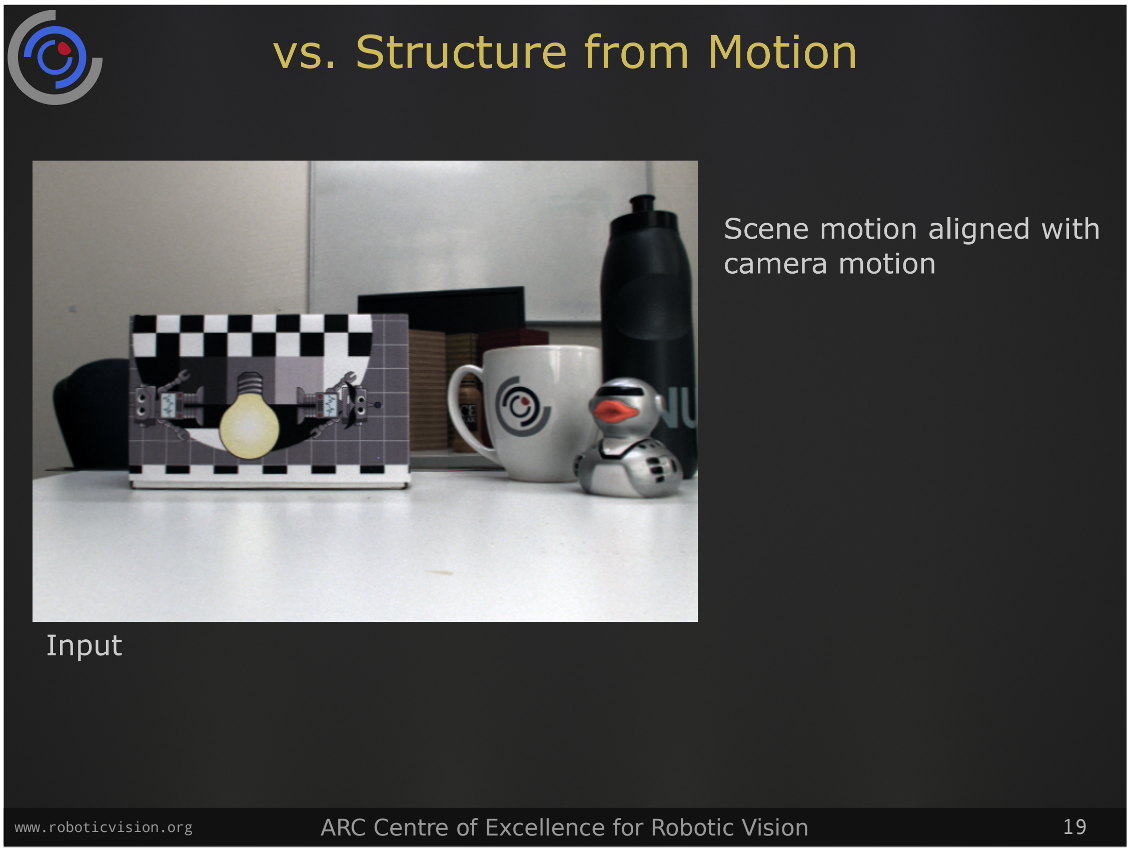
Input

Scene motion aligned with camera motion

This isn't mentioned in the abstract
Let's compare with an SfM approach

SfM is used to align two views, and the error taken to effect change detection

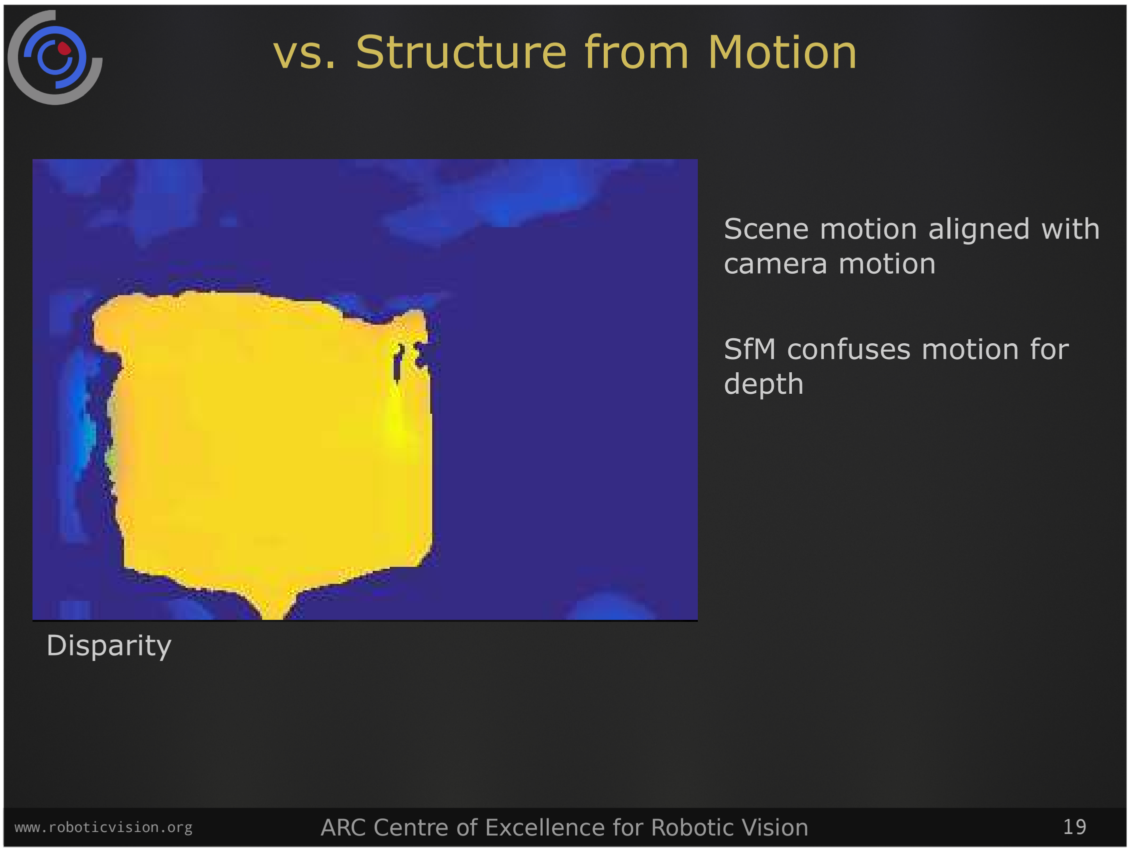
What if the motion is parallel with camera motion?



This isn't mentioned in the abstract
Let's compare with an SfM approach

SfM is used to align two views, and the error taken to effect change detection

What if the motion is parallel with camera motion?



SfM confuses the motion for depth – disparity should not be so high around the box

Here stereo is standing in for SfM.

Because it's doing the same thing with fewer unknowns, stereo should represent an upper bound on SfM's performance

vs. Structure from Motion

Actual change

Scene motion aligned with camera motion

SfM confuses motion for depth

Ground truth = temporal derivative

vs. Structure from Motion

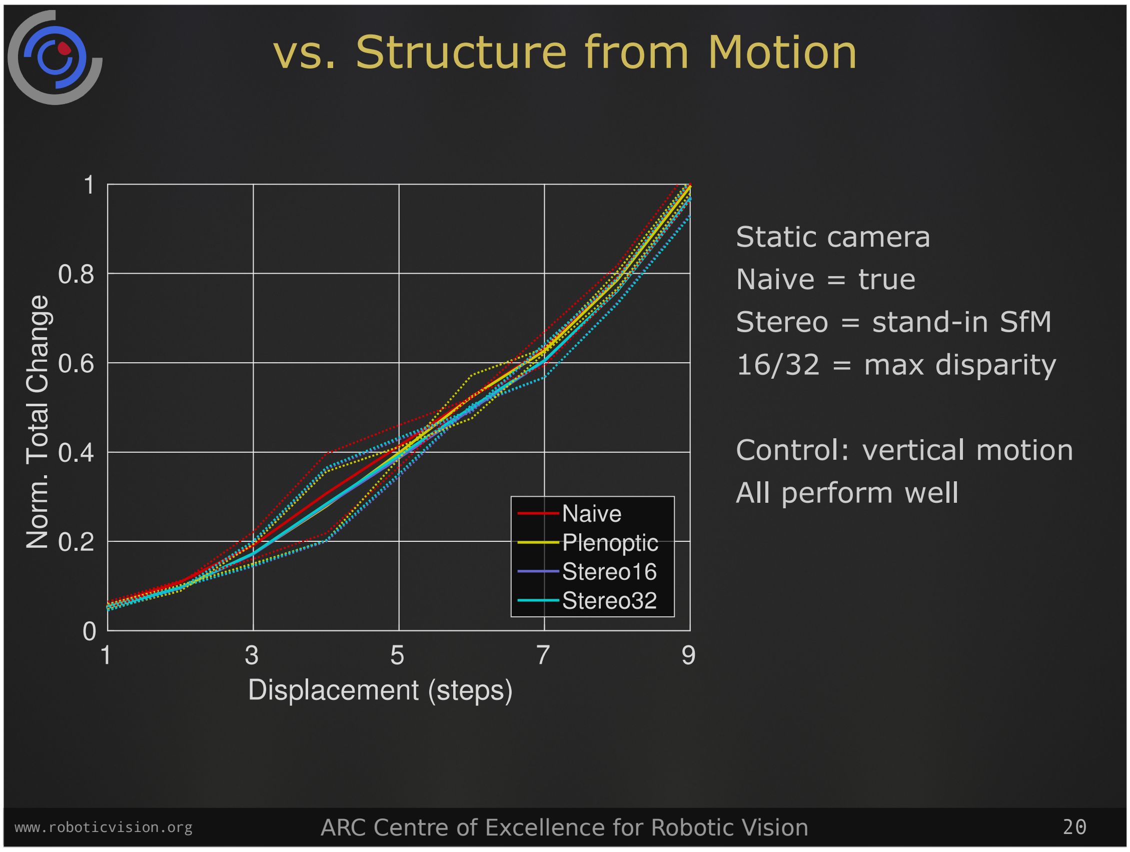
SfM-based change estimate

Scene motion aligned with camera motion

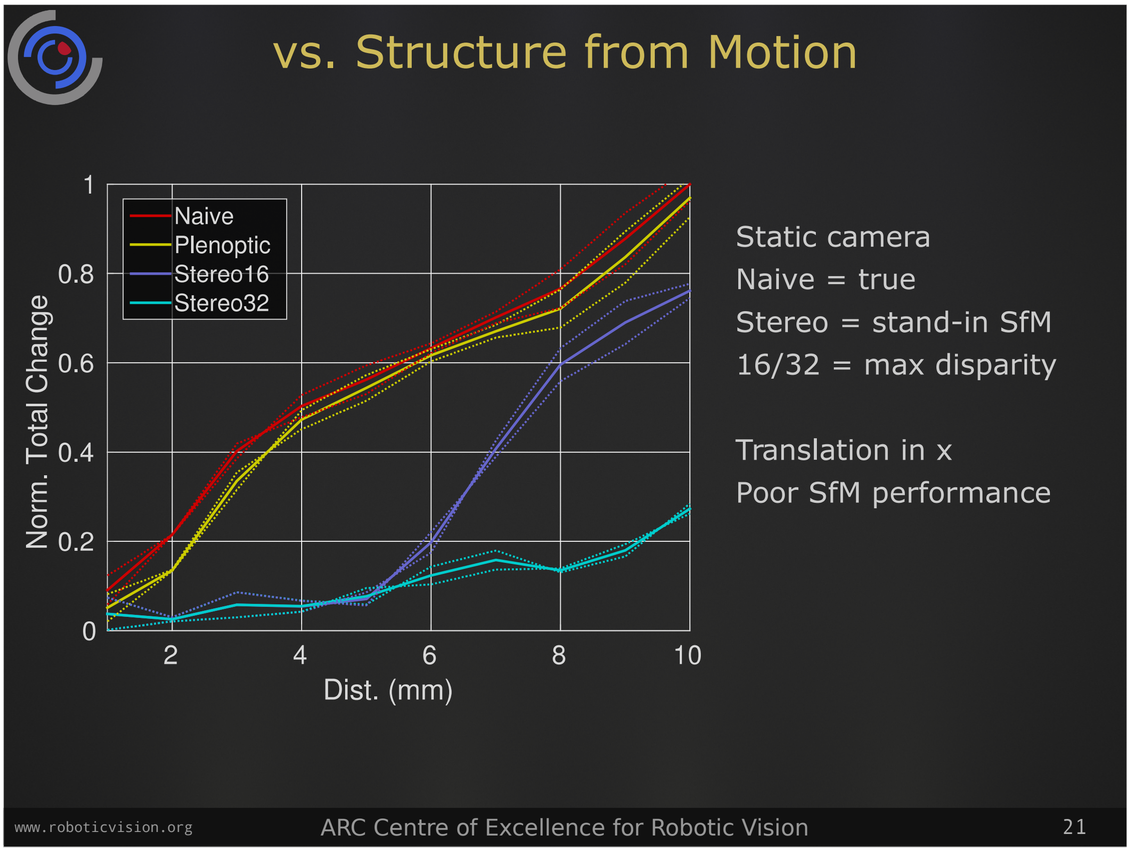
SfM confuses motion for depth

Poor change detection

SfM (stereo) change estimate is poor (should look like temporal derivative, note massive hole in middle)

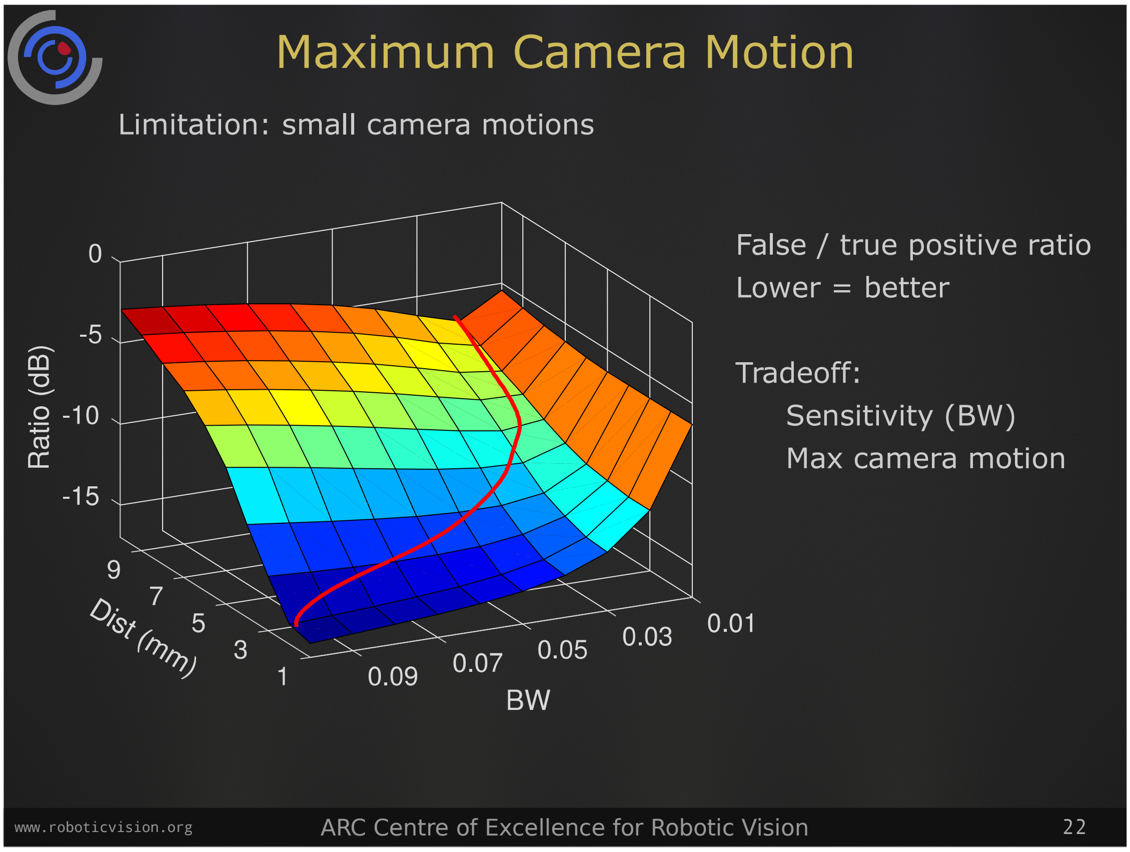


Quantifying the SfM breakdown, this is a control to show that it works when motion isn't aligned with camera motion



Motion here is parallel with camera motion, making the SfM method suffer

The shift in performance for stereo16 is where motion exceeded 16 pixels, saturating the disparity estimate



Common question is how small camera motion has to be. This answers it.

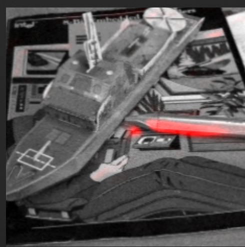
BW is the input bandwidth: a smoothing filter is used to increase coherence, but eventually this makes things less sensitive to change

This result is for a dynamic scene, with hand-labelled ground truth

If you turn the bandwidth too low, you lose sensitivity to changes (right side of the plot)

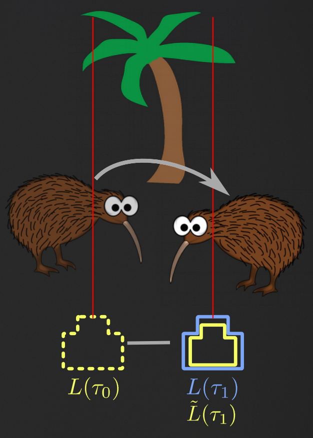
Conclusions

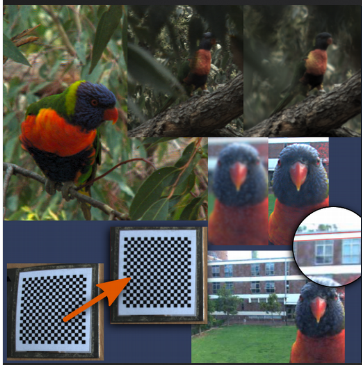
- Simplified change detection for moving cameras
  - Closed-form, simple, parallel
  - Outperforms monocular SfM for common scenes
  - Limited camera motion, tradeoff with sensitivity
- Framework to simplify other problems
  - Moving camera → Virtual static camera



What's Next?

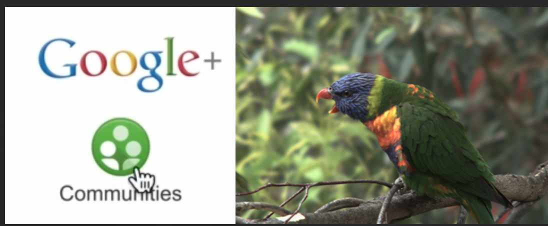
- Other still-camera solutions
  - Object tracking, segmentation, isolation and removal, denoising, velocity & temporal filtering
- Better imaging
  - Custom cameras, hybrids
- Simplify difficult tasks
  - 4D algorithms, sensor fusion





Light Field Toolbox for MATLAB

- Load Gantry and Lytro imagery
- Calibrate and rectify Lytro imagery
- Linear depth, volume filters
- Denoising: low-light, fog, dust, murky water
- Occluder removal: rain, snow, silty water



ACRV
AUSTRALIAN CENTRE FOR ROBOTIC VISION

ACFR
AUSTRALIAN CENTRE FOR FIELD ROBOTICS

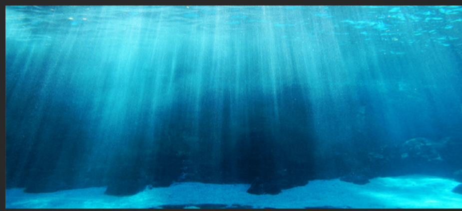
QUT
Queensland University of Technology
Brisbane Australia

THE UNIVERSITY OF SYDNEY

Australian Government
Australian Research Council

Challenges in Robotic Vision

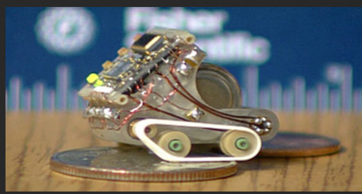
Environment



- Variable light: Day & night
- Weather
- Participating media
- Unstructured, dynamic scenes

Challenges in Robotic Vision

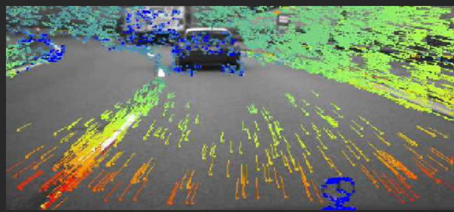
Platform



- Power, computing
- Weight, volume
- Actuation
- Time

Challenges in Robotic Vision

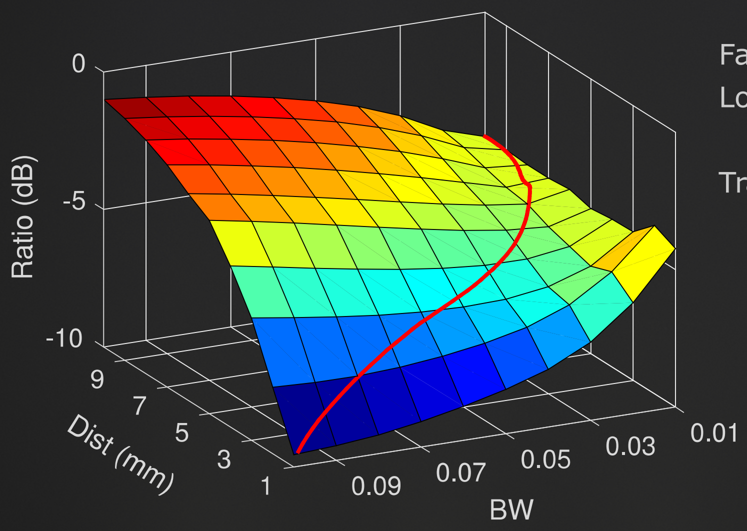
Camera



- Light vs. depth of field
- Light vs. motion blur
- Nonunifom apparent motion

Maximum Camera Motion

Limitation: small camera motions, static scene



False / true positive ratio

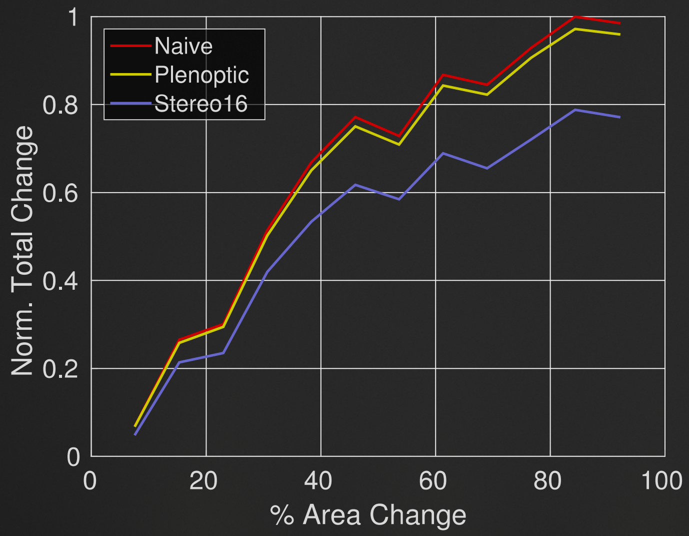
Lower = better

Tradeoff:

- Sensitivity (BW)
- Max camera motion

Maximum Scene Motion

Limitation: what if the whole scene is moving?

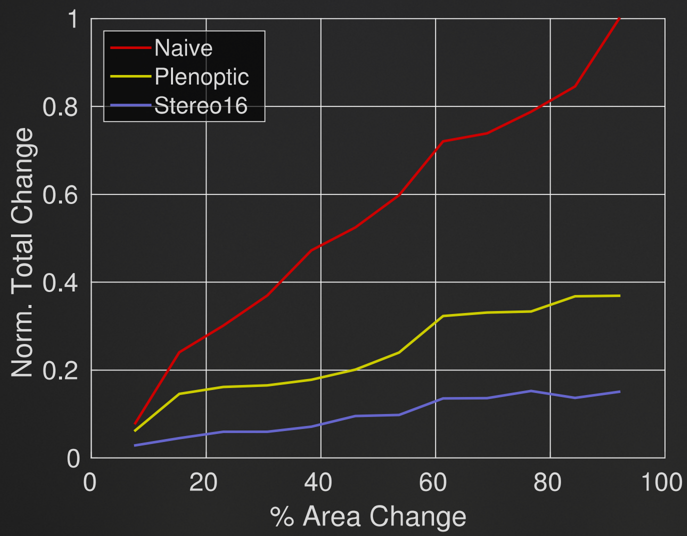


Static camera
Naive = true

- Small motions
- Random, incoherent
- Good performance

Maximum Scene Motion

Limitation: what if the whole scene is moving?

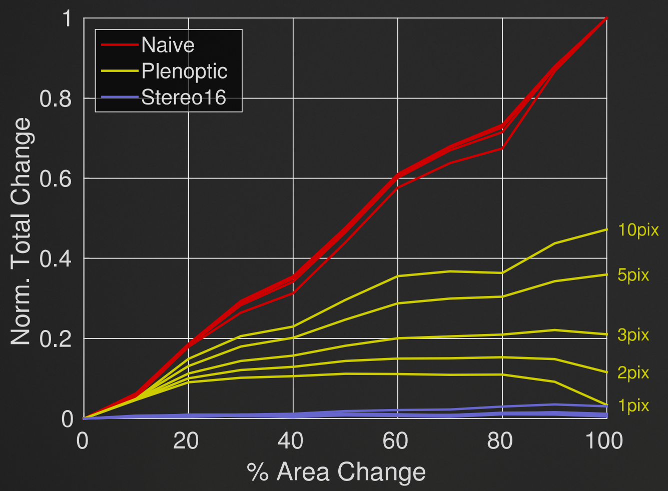


Static camera
Naive = true

- Small motions
- Coherent, appears as apparent motion
- Poor performance

Maximum Scene Motion

Limitation: what if the whole scene is moving?

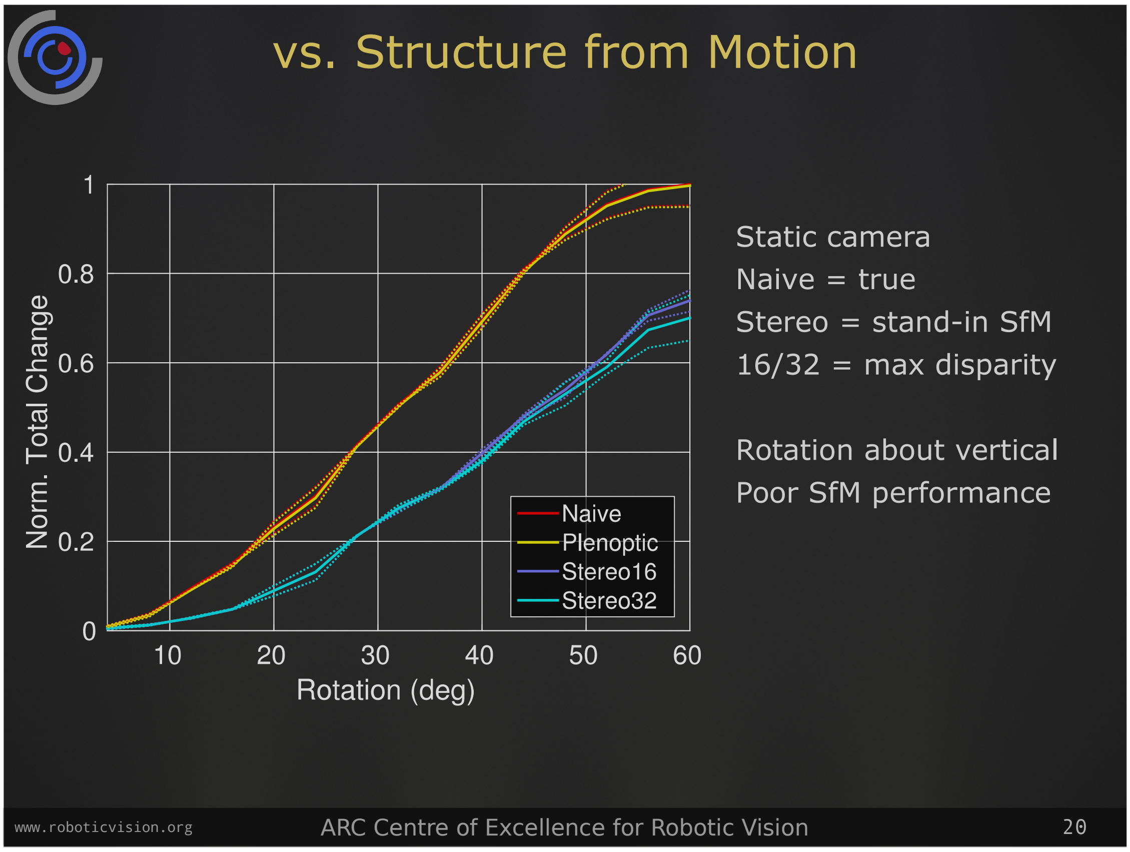


Static camera

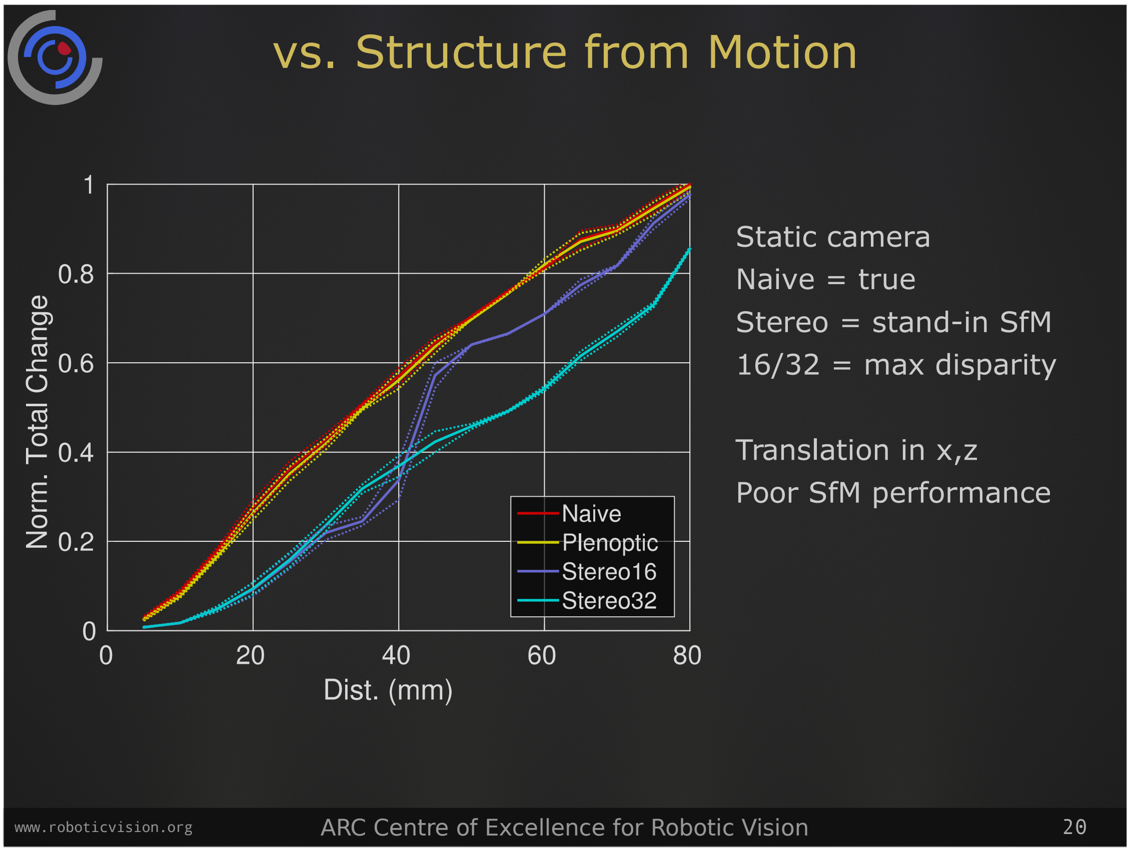
Naive = true

Simulation to find worst performance:

- Small motion
- Across whole image
- Coherent, appears as apparent motion



Here motion is parallel with camera motion, so the stereo methods suffer



The shift in performance is where motion exceeded 16 pixels, saturating stereo16's disparity estimate